Grid Enabled Optimisation Using Evolutionary Algorithms

Thesis submitted for candidature for the degree of PhD

Alex Shenfield

January 2008

Department of Automatic Control and Systems Engineering

The University of Sheffield

Summary

Optimisation and decision support tools are vital in all areas of engineering. Many engineering design problems, from the design of a controller for aircraft stability to the development of automobile chassis, can be effectively addressed by using evolutionary algorithms to optimise computational models of the systems under consideration. Unfortunately, for non-trivial problems, capturing the dynamics of a system with high fidelity often results in a model that is very computationally expensive. However, this level of fidelity is needed for an engineer to have confidence in the final solutions produced by an optimiser.

Evolutionary algorithms aggravate this problem by often requiring many thousands of candidate solutions to be evaluated, since they search a population of points, before finding a satisfactory final solution. However, evolutionary algorithms do exhibit a large degree of parallelism, making them well suited for exploiting the emerging paradigm of Grid computing in which engineers and scientists have transparent access to large amounts of compute resources 'on demand'.

In this work, strategies to accelerate the process of optimisation in engineering design are investigated. One promising approach examined in this thesis is the use of computational Grids in engineering design optimisation to reduce the time needed to obtain useful results. This allows the optimisation of much higher fidelity models than was previously possible. Another promising strategy is in the computational steering of multi-objective evolutionary search methods, where decision making is closely integrated into the search process resulting in a decision maker having finer control over the algorithm than was possible before.

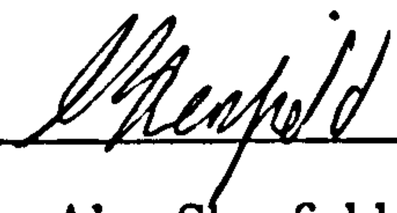
Acknowledgments

I'd like to thank my parents for their love, support and encouragement in everything I've ever done. I'd also like to thank Megan, without whose support and patience I'd never have got this far. I love you very much, and am thankful every day that we're together.

I would also like to express my sincere thanks to my supervisor, Professor Peter Fleming, for the opportunity to do postgraduate research, and for the excellent advice and encouragement he's provided along the way. I'm extremely grateful to all the people that I've had the pleasure of working with, in particular Shahid, Luka, Max, Xiaoxu, and Andy; and to all my friends, who have been fantastically patient with me (over the last couple of years especially). Finally I'd like to thank the UK Engineering and Physical Sciences Research Council for their financial support.

Statement of Originality

Unless otherwise stated in the text, the work described in this thesis was carried out solely by the candidate. None of this work has already been accepted for any other degree, nor is it being concurrently submitted in candidature for any degree.

Candidate: 

Alex Shenfield

Supervisor: 

Peter J. Fleming

For Megan.

I love you enough to melt all the tigers in the world to butter.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Outline of Thesis 2

1.3 Contributions 4

2 Review of EMO **6**

2.1 Introduction 6

2.2 Evolutionary Computation 7

2.2.1 A Generic Evolutionary Algorithm 8

2.2.2 Further Concepts in Evolutionary Computation 16

2.2.3 Advantages and Disadvantages of EC 24

2.2.4 Applications of Evolutionary Computation 25

2.3 Multi-Objective Optimisation 28

2.3.1 Pareto-Dominance and Optimality 28

2.3.2 An Ideal Multi-Objective Optimiser 30

2.3.3 Classical Multi-Objective Optimisation Methods 32

2.4 Evolutionary Multi-Objective Optimisation 33

2.4.1 Advantages of the Evolutionary Approach 33

2.4.2 Evolutionary Multi-Objective Optimisation:

A History 34

2.4.3 Obtaining Good Proximity 39

2.4.4 Obtaining Good Diversity 41
2.4.5 Obtaining Good Pertinency 42
2.4.6 Applications of Evolutionary Multi-Objective Optimisation 50
2.5 Summary 52

3 Review of Grid Computing 54
3.1 Introduction 54
3.2 History of Grid Computing 56
3.2.1 Networking History 57
3.2.2 Supercomputing History 59
3.3 Services for Application Development on the Grid 63
3.3.1 Resource Monitoring and Discovery 66
3.3.2 Secure Authentication 68
3.3.3 Execution Management 69
3.3.4 Data Management 70
3.4 Applications of Grid Computing 72
3.4.1 The Synthetic Forces Express Project 72
3.4.2 The National Fusion Collaboratory Project 73
3.4.3 The Distributed Aircraft Maintenance Environment Project 75
3.5 Summary 77

4 Grid-based Scheduling 79
4.1 Introduction 79
4.2 Survey of Resource Management Systems 81
4.2.1 Resource Management Systems with Centralised Control 82
4.2.2 Meta-Scheduling Architectures for Grid Computing 84
4.3 The White Rose Grid 87

4.4 Design Patterns for Meta-Schedulers 88
4.5 Workload Allocation Strategies 100
4.5.1 Introduction 100
4.5.2 Modelling the System 102
4.5.3 Workload Allocation Schemes for Bulk Arrivals 103
4.5.4 Performance Evaluation of Workload Allocation Algorithms 106
4.6 Introducing a Novel Hybrid Approach to Job Scheduling 114
4.7 Summary 118

5 Grid-Enabled Optimisation **120**
5.1 Introduction 120
5.2 Parallel Evolutionary Optimisation 121
5.2.1 Introduction to Parallel Evolutionary Algorithms and Population Topologies 121
5.2.2 Related Work 125
5.3 Comparison of the Island and Global Models of Parallel EC . . . 127
5.3.1 Experimental Set-Up 127
5.3.2 Single-Objective Performance Comparison 129
5.3.3 Multi-Objective Performance Comparison 135
5.4 Introducing a Grid Based Framework for Evolutionary Optimisation 141
5.4.1 Parallelisation of the Evolutionary Algorithm 141
5.4.2 Optimisation in a Service-Oriented Architecture 142
5.5 Application of the Evolutionary Optimisation Framework to Real World Problems 144
5.5.1 An Aero-Engine Maintenance Scheduling Example 145
5.5.2 An Aircraft Control Systems Design Example 153
5.6 Summary 162

6 Computational Steering of EMO **167**

6.1 Introduction 167

6.2 Computational Steering 169

6.3 Decision Making in Engineering Design 171

6.4 Implementation of a Computational Steering System for Multi-Objective Optimisation 173

6.4.1 Steering of the Multi-Objective Evolutionary Algorithm . . 173

6.4.2 Visualisation 176

6.4.3 A PDA Implementation 181

6.5 Application of a Computational Steering System for Multi-Objective Optimisation 182

6.6 Summary 190

7 Conclusions **193**

7.1 Grid Computing in Science and Engineering 194

7.2 Parallel Evolutionary Computation on the Grid 195

7.3 Computational Steering of Evolutionary Multi-Objective Optimisation 195

7.4 Future Challenges 196

7.4.1 Running Applications on the Grid 196

7.4.2 Evolutionary Computation 197

A The M/M/1 Queue With Bulk Arrivals **199**

B Rearranging Equation 4.6 **204**

C Solving Equation 4.9 **206**

References **209**

Acronyms	
CPU	Central Processing Unit
DE	Differential Evolution
DLL	Dynamic Least Load
DM	Decision Maker
EA	Evolutionary Algorithm
EC	Evolutionary Computation
EMO	Evolutionary Multiobjective Optimisation
EP	Evolutionary Programming
ES	Evolution Strategies
GA	Genetic Algorithm
GB	Gigabyte
GGF	Global Grid Forum
GRAM	Grid Resource and Allocation Management
GSI	Grid Security Infrastructure
HPC	High Performance Computing
HTC	High Throughput Computing
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
MCDM	Multi Criteria Decision Making
MDS	Monitoring and Discovery System
MEAROS	Modular Engine Arisings, Repair and Overhaul System
MOEA	Multi Objective Evolutionary Algorithm
MPP	Massively Parallel Processing
OGSA	Open Grid Services Architecture
OMR	Optimised mean Miss Rate
ORT	Optimised mean Response Time
OS	Operating System
PB	Petabyte
PDA	Portable Digital Assistant
RMS	Resource Management System
ROI	Region Of Interest
SMP	Symmetric Multiple Processor
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TFLOPS	TeraFLOPS (One Trillion Floating Point Operations Per Second)
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
VO	Virtual Organisation
W3C	World Wide Web Consortium
WSDL	Web Services Definition Language
XML	eXtensible Markup Language

Table 1: Table of Commonly Used Acronyms

Chapter 1

Introduction

1.1 Motivation

Problems arise in every aspect of life, from planning a summer holiday to designing an engine control unit (ECU) for a car. Some of these problems are easy to solve, but many provide more of a challenge. Polya (1957) outlines four general steps for solving problems:

1. Understand the problem.

2. Make a plan.

3. Carry out this plan.

4. Assess the solution produced.

It is important to not only understand the problem, but also to understand the underlying assumptions in the problem solving process. As Michalewicz and Fogel (2000) point out, frequently when an engineer solves a problem in the real world they are really just finding the solution to a simplified model of the problem. For the engineer to have confidence in the final solution produced by the problem solving process it is important that the model captures all the relevant dynamics of the true problem.

Unfortunately many conventional problem solving techniques have difficulty when dealing with complex models. This leaves two choices:

1. Simplify the model so as to be solvable with conventional techniques and risk obtaining a solution that is sub-optimal with respect to the true problem.

2. Use non-traditional methods to solve the more complex, higher fidelity model.

Problems within the engineering design domain are typically complex and difficult to solve, frequently requiring the simultaneous consideration of multiple performance criteria. It is therefore essential that engineers have access to advanced modelling, optimisation and decision support techniques to assist them in arriving at satisfactory solutions. Soft computing methods inspired by nature, such as artificial neural networks and evolutionary computation, provide a powerful and versatile set of tools to assist in tackling such problems in engineering design; however, this power and versatility comes at a price - with such techniques often being computationally expensive and potentially taking hours or days to arrive at a solution.

The computationally expensive nature of these soft computing techniques has led to increasing interest - in the engineering design domain especially - in accelerating this problem solving process. This thesis will explore two possible ways of accomplishing this goal: firstly, by utilising the recent paradigm of Grid computing to assist engineers in tackling engineering design problems; and secondly, by improving the integration of decision making in the problem solving process.

1.2 Outline of Thesis

Chapter 2 presents a thorough review of the state of the art in evolutionary computing and multi-objective optimisation, including theoretical concepts such

as Pareto optimality. A brief history of the development of evolutionary multi-objective optimisation methods is given, and some of the key issues in developing effective MOEAs are examined.

Chapter 3 presents a review of the Grid computing paradigm starting from a historical perspective and progressing to the current state of the art. It aims to show the reader how Grid computing differs from conventional distributed computing, before presenting some real-world examples of the Grid being used to enable science and engineering on a larger scale than previously possible.

Chapter 4 examines resource management in computational Grid environments and provides a design pattern based architecture for the construction of application-centric meta-schedulers. These design patterns encapsulate a proven solution for the construction of resource brokers for Grid computing environments and constitute a step towards maturity for the Grid computing paradigm. Further issues in scheduling jobs in a multi-institutional computational Grid are examined and a novel workload allocation algorithm and job scheduling strategy is developed for the optimal allocation of batches of jobs amongst available resources.

Chapter 5 explores the coupling of evolutionary optimisation methods with the Grid computing paradigm to assist in solving complex engineering design problems. A key issue in the parallelisation of evolutionary algorithms is the choice of parallelism to use, and a major contribution of Chapter 5 is a thorough investigation into the suitability of these parallel EA models to Grid computing environments. A flexible Grid enabled framework for evolutionary optimisation of computationally expensive real-world engineering design problems is also introduced and guidelines for its use proposed.

Chapter 6 considers the effects that computational steering of evolutionary multi-objective optimisation can have on the solution set produced by an optimiser. A computational steering system for multi-objective evolutionary algorithms that integrates decision making and optimisation is developed, and

its efficiency on an engineering design problem with many criteria is analysed.

Finally, Chapter 7 presents some conclusions that can be drawn from the work presented in this thesis and outlines some areas that merit further investigation.

1.3 Contributions

The main contributions of this thesis are:

- **Development of a novel Grid based resource broker.** A novel workload allocation algorithm and job scheduling strategy for batches of jobs arriving simultaneously has been developed and shown to outperform standard approaches from the literature.

- **A rigorous performance comparison between the predominant models for parallel evolutionary computation.** This comparison uses established statistical methods to compare the performance of these models on a variety of single and multi-objective test problems that are commonly used in the literature.

- **Development of a Grid-enabled framework for evolutionary optimisation.** This framework, the development of which was motivated by the desire to solve complex problems in engineering design, has been used to solve two computationally expensive real-world problems. This work has been published in Shenfield and Fleming (2005) and Shenfield, Fleming, Allan and Kadirkamanathan (2007).

- **Development of a computational steering system for evolutionary multi-objective optimisation.** This steering system provides a close integration between decision making and search and is successfully used to tackle a many criteria engineering design problem. This work has been published in Shenfield, Fleming and Alkarouri (2007).

Additional contributions that have arisen from this work but are not specifically included in this thesis are:

- **An overview of modelling, optimisation and decision support in the Distributed Aircraft Maintenance Environment (DAME).** This report provides an overview of the integrated set of tools developed at the University of Sheffield as part of the £3M DAME project. This work was published in Shenfield *et al.* (2005).

- **Enhancement of the Genetic Algorithm Toolbox for MATLAB.** The GA Toolbox (Chipperfield *et al.* 1994) developed at the University of Sheffield has been updated to work under MATLAB version 6 onwards, and additional evolutionary operators have been developed during the course of this research.

Chapter 2

Review of Evolutionary Multi-Objective Optimisation

2.1 Introduction

This Chapter aims to provide a thorough review of the field of evolutionary computation, with an emphasis towards its use in solving multi-objective problems. Section 2.2 provides a general introduction to evolutionary computation: first introducing a generic evolutionary algorithm, before then discussing some more advanced evolutionary computing concepts and the advantages and disadvantages of evolutionary optimisation techniques.

Multi-objective optimisation is introduced in Section 2.3, with some explanation of the theoretical concepts and some discussion of the aims of multi-objective optimisation. Section 2.4 then reviews the use of evolutionary methods in solving multi-objective problems: first outlining the advantages that evolutionary techniques provide, and then providing a historical perspective. Some strategies that help evolutionary methods achieve the goals of the multi-objective optimisation process are also discussed.

2.2 Evolutionary Computation

Evolutionary Computation (EC) refers to a set of optimisation and search methods inspired by the process of natural selection (Darwin 1859) and population genetics (Fisher 1930). EC is a sub-discipline of the field of soft computing, which also includes other techniques inspired by nature such as neural networks. Initially much of the interest in EC was directed at using EC to study natural and economic processes (Barricelli 1962, Holland 1975). However, EC has proved to be extremely useful in solving complex optimisation problems, especially those in areas where conventional techniques struggle (such as those with discontinuous and multi-modal search spaces).

The roots of evolutionary computation can be traced back to the late 1950s with the work of Friedberg (Friedberg 1958, Friedberg *et al.* 1959) and Box (1957). However, it wasn't until the development of more powerful computer platforms in the 1970s that evolutionary computation started to advance. Contemporary EC methods have their origins in three independently developed, but related, approaches: the German *Evolutionstrategian* (Evolution Strategies, ESs) developed by Rechenberg (1973), and the American *Evolutionary Programming* (EP) and *Genetic Algorithms* (GAs) developed by Fogel *et al.* (1966) and Holland (1975) respectively.

Each of the above methods were developed to address certain problems, and as a consequence they emphasise different aspects of evolutionary computation. For example, Evolution Strategies were initially developed to solve experimental optimisation problems with mainly continuous parameters (Bäck *et al.* 1991), while the original goal of Evolutionary Programming was to create artificial intelligence by evolving Finite State Machines (FSMs) to predict events on the basis of previous observations (Bäck *et al.* 1997). EP was then extended to work with continuous variables. Initially developed as a general model for adaptive processes, Genetic Algorithms differ from the other two approaches in their use

of a discrete representation of decision variables.

However, over the last three decades, as the different EC communities have exchanged ideas, the distinctions between the different approaches have weakened. Researchers in evolutionary computation are now encouraged to think in broader terms (Michalewicz and Fogel 2000), and the community as a whole has adopted the term *Evolutionary Algorithm* (EA) to refer to a generic evolutionary search.

2.2.1 A Generic Evolutionary Algorithm

Evolutionary Algorithms are a stochastic, population based optimisation method utilising some of the mechanisms of natural selection (Goldberg 1989). EAs use computational models of evolutionary operators such as reproduction, mutation and selection to progressively improve a population of candidate solutions. Figure 2.1 shows the general structure of an evolutionary algorithm.

```
Procedure EvolutionaryAlgorithm
begin
    gen = 0
    initialise P(gen)
    evaluate P(gen)
    while not finished do
    begin
        gen = gen + 1
        mating_selection:
            Select P'(gen) from P(gen-1)
        vary P'(gen)
        evaluate P'(gen)
        environmental_selection:
            Select P(gen) from P'(gen) and P(gen-1)
    end
end
```

Figure 2.1: A Pseudo-Code Implementation Of an Evolutionary Algorithm

As Figure 2.1 shows, the population of candidate solutions in an EA is evolved over successive generations by selection and random variation. Variation provides a means of exploring the search space to discover new solutions, whereas selection ensures that good solutions found by the algorithm are exploited when producing the following generation (see Section 2.2.2 for more detail on exploration and exploitation in EAs). The general form of an evolutionary algorithm presented here uses two different selection operators: *mating selection* and *environmental selection* (often termed *selection-for-variation* and *selection-for-survival* in evolutionary biology). The EA operators are described in more detail in the following sections.

Population

Evolutionary Algorithms manipulate an encoded representation of the candidate solutions to a problem. Classical genetic algorithms have traditionally used a binary string representation to encode each of the decision variables that form a candidate solution. This emphasis on the use of binary strings stems from fundamental GA theory (Holland 1975, Goldberg 1989), which suggests that alphabets with low cardinality result in enhanced schema processing (Goldberg 1991). However, Michalewicz (1996) has shown that this representation is inappropriate for some problems (such as those with multi-dimensional, high-precision, numerical search spaces). In contrast to traditional GAs, other evolutionary approaches (such as Evolution Strategies) have focussed more on the use of continuous variables to represent candidate solutions.

Fogel and Ghoziel (1997) have shown that there is no intrinsic advantage in choosing one bijective representation over another, although particular representations may be more computationally tractable or efficient for certain problems. Equivalent evolutionary operators can be created regardless of the representation used. Consequently modern EA practice emphasises choosing a representation that is appropriate for the problem under consideration. For

example, evolutionary algorithms solving real-valued parameter optimisation problems generally use a real-valued representation (Herrara *et al.* 1998). Deb and Goyal (1996) have even proposed a mixed representation for optimisation of engineering design problems such as the design of a cantilever beam to carry a certain load.

Evaluation

The evaluation function of an evolutionary algorithm assigns a *cost* value to each candidate solution based on its performance. It is important that the evaluation function accurately captures the dynamics of the true problem. This mean that not only must the global optima of the evaluation function correspond to that of the problem it is representing, but that the evaluation function should also provide enough information to differentiate between good solutions and poor ones.

After each candidate solution has been evaluated, a *fitness* value is derived from its cost (taking into account the other members of the population). This fitness value is used in the mating selection operator (described later) and, while the cost value of each individual is a property of the problem, the fitness assignment strategy can be easily altered in the algorithm. The main fitness assignment methods are *proportional fitness assignment* (Holland 1975), where the fitness of the individual is assigned as some function of the cost values, and *rank-based fitness assignment* (Baker 1985), where the fitness is assigned according to the rank of the individual in the population. Rank-based fitness assignment has been shown by Whitley (1989) to overcome many of the difficulties associated with proportional fitness assignment methods (such as super-fit individuals dominating the population).

Mating Selection

The purpose of mating selection is to choose a set of individuals, $P'(gen)$, from the current generation to which variation operators are applied to form the next generation. Mating selection uses the fitness values (see above) to form the set of individuals to be varied. The 'fitter' the individual, the higher chance it has of contributing to the next generation. It is worth noting, however, that mating selection is often performed stochastically to avoid systematic errors in the selection process, and thus there is no guarantee that the individual with the highest fitness will be chosen to contribute to the next generation.

An ideal mating selection scheme must retain a constant population size whilst attempting to provide accurate, consistent and efficient sampling of the population (Baker 1987). The accuracy of the sampling in a mating selection scheme is measured in terms of the bias (the absolute difference between the actual sampling probability of an individual and the expected value), and the consistency of the sampling is measured in terms of the spread (the maximum deviation of the number of offspring produced by an individual from the expected value). To ensure accurate and consistent sampling, both the bias and the spread should be minimised. An efficient mating selection scheme should not increase the time complexity of the EA and, ideally, the selection scheme should be parallelisable.

Many selection schemes have been proposed in the literature, and the interested reader is directed to Blickle and Thiele (1995) or Goldberg and Deb (1991) for a thorough review and analysis. Three commonly used selection methods are *Roulette Wheel Sampling* (RWS, or sampling with replacement) (Goldberg 1989), *Stochastic Universal Sampling* (SUS) (Baker 1987), and *Tournament Selection.* Stochastic Universal Sampling guarantees sampling with zero bias and minimum spread, and is generally considered superior to Roulette Wheel Sampling and Tournament Selection (Hancock 1994). However, Tournament Selection can provide some advantages when dealing with noisy evaluation functions (Miller

and Goldberg 1995).

Variation

The purpose of variation in evolution is simply to introduce change into the population. Without variation, the evolutionary process would stagnate at the best individual in the initial generation. In evolutionary search, variation acts to explore the search space. In general, the choice of variation operators is dependant on the representation used (see previously). Variation operators can be broadly divided into *mutation* and *recombination.*

Mutation operates on a single parent individual and makes changes to the chromosome of that individual with some probability. A classic example is 'bit-flipping' mutation in the traditional GA, where each 'bit' in the genotype is 'flipped' with some probability (see Figure 2.2). This change has traditionally been considered a secondary mechanism of genetic algorithm adaptation (after recombination) and typically occurs with a small probability (Goldberg 1989). In evolution strategies mutation plays a much greater rôle, with early ESs using only mutation and selection as their evolutionary mechanisms (Bäck *et al.* 1991). In this approach mutation is achieved by adding normally distributed random numbers to the parent individuals (see Figure 2.3), in accordance with the observation that smaller changes occur more frequently in nature than larger ones (Michalewicz 1996). ESs often use a *self-adaptive* approach in setting the standard deviation of the mutation operator, adapting the standard deviation parameter in parallel to the evolution of the candidate solution.

Recombination (also know as crossover) operates on two or more parent individuals. It acts to combine genetic material from multiple parents to form new offspring. The simplest example of this is the *single-point binary crossover* used in the traditional GA (Holland 1975, Goldberg 1989). In this form of recombination two parents form two new individuals (termed *offspring*) by swapping all their genetic material after a randomly chosen point on the chromosome (see Figure

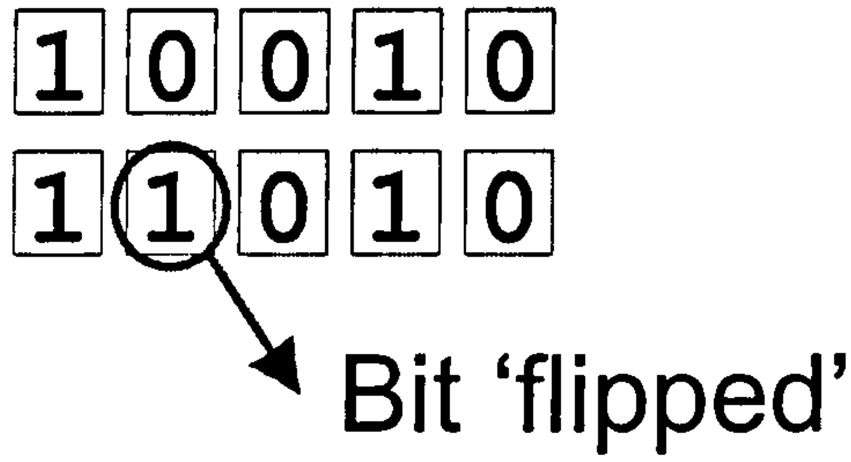


Figure 2.2: Bit Flipping Mutation in a Binary-Coded Evolutionary Algorithm

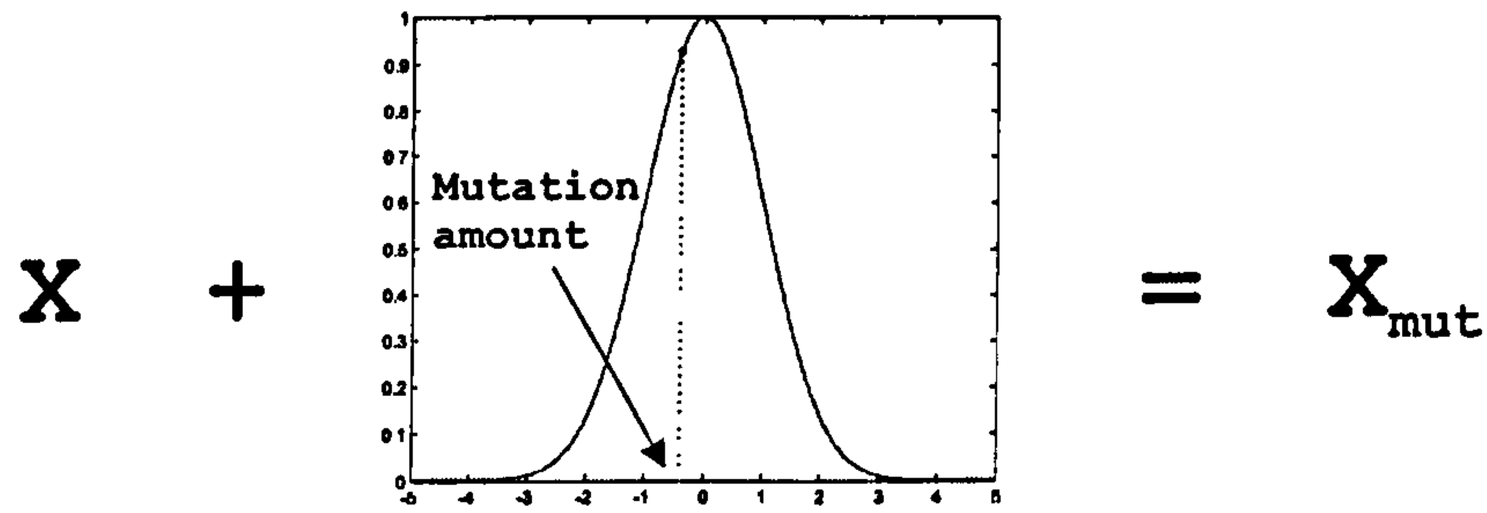


Figure 2.3: Gaussian Mutation in a Real-Coded Evolutionary Algorithm

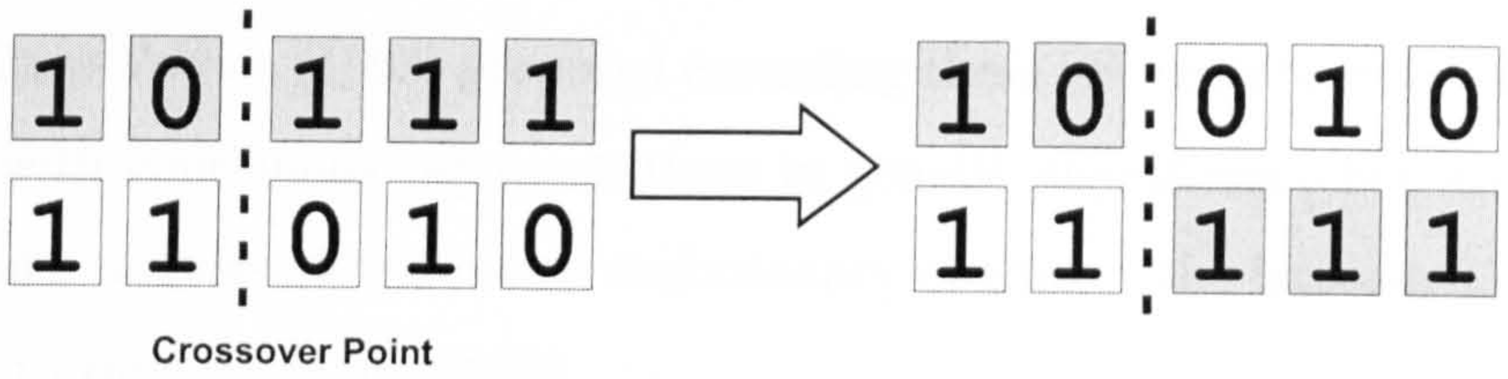


Figure 2.4: Single Point Binary Crossover in a Binary-Coded Evolutionary Algorithm

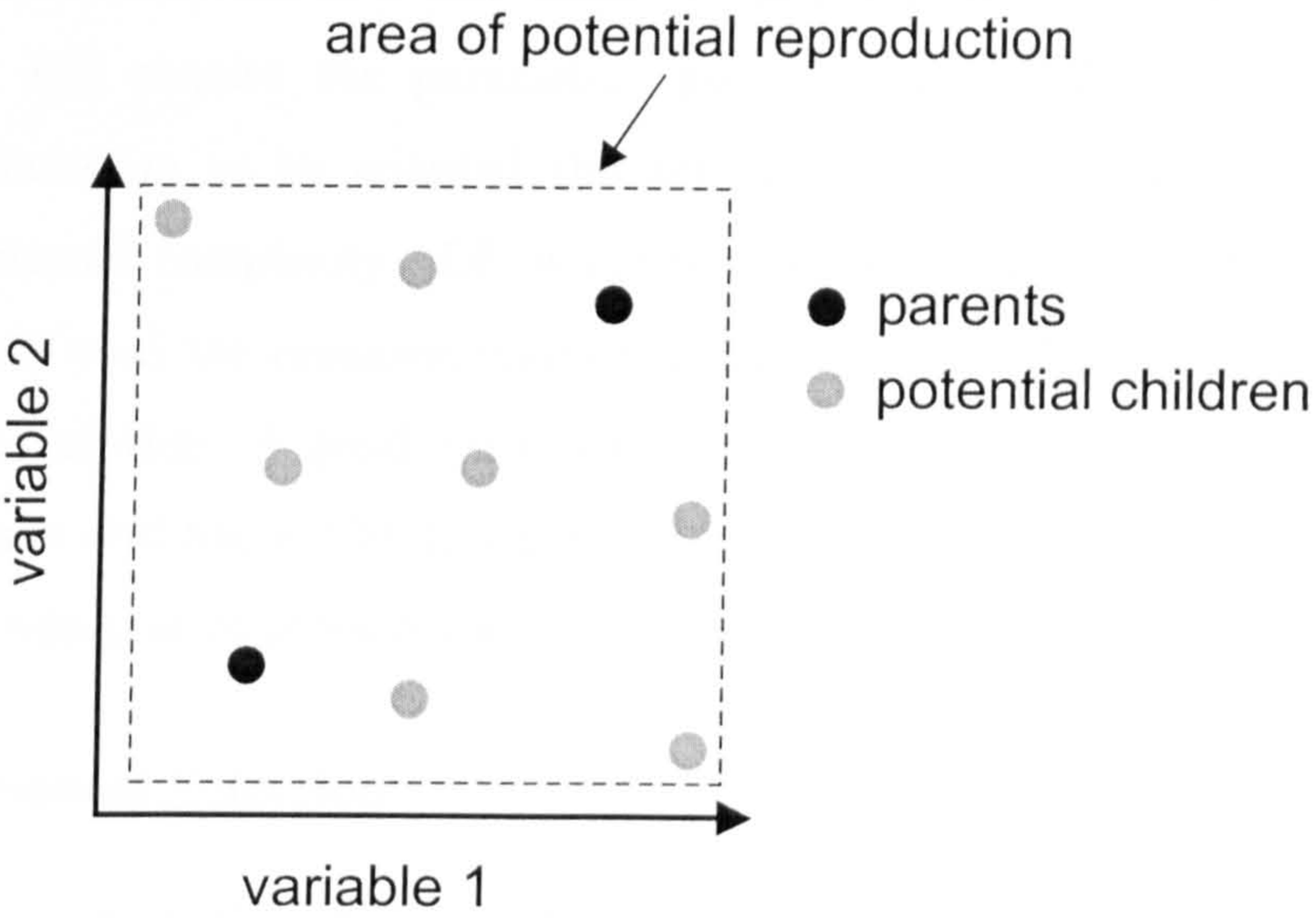


Figure 2.5: Extended Intermediate Recombination in a Real-Coded Evolutionary Algorithm

2.4). This technique has been extended to a *multi-site binary crossover* operator, where the genetic material between multiple sites on the chromosome is swapped (De Jong 1975). Recombination in genetic algorithms is typically applied with a much higher probability than mutation.

Although early ES approaches relied solely on mutation for variation of the population, recombination operators (such as intermediate recombination, see

Figure 2.5, and discrete recombination) were introduced later. Mühlenbein and Schlierkamp-Voosen (1993) proposed extending these ES recombination operators to deal with continuous representations in genetic algorithms. For a thorough review and analysis of real-coded evolutionary operators the interested reader is directed to Herrara *et al.* (1998).

Differential Evolution (DE) (Price and Storn 1995) is unusual amongst EC strategies in using a single combined crossover and mutation operator. DE, like evolution strategies, uses a floating point representation to encode candidate solutions and so is well suited to real-parameter optimisation problems. However, ESs require the parameters governing the probability distribution used in mutation to be adapted throughout the optimisation process, which adds additional complexity. DE is entirely self-adapting, since it deduces the information used for crossover/mutation directly from the distribution of the existing population. A good introduction to differential evolution can be found in Lampinen and Storn (2004) where DE is explained and shown to effectively address a selection of problems in engineering design.

Environmental Selection

Typically the population size of an evolutionary algorithm remains constant over its run. However, after the application of the variation operators there exists two sets of solutions: the parent generation, $P(gen)$, and the child generation, $P'(gen)$. As more solutions now exist than can be used in the next generation, $P(gen+1)$, the algorithm needs to decide which solutions to *reinsert* into the population to form the next generation. This reinsertion process is often termed *environmental selection.*

Evolution strategies use the notation $(\mu + \lambda)$ and (μ, λ) to describe possible environmental selection schemes (Beyer and Schwefel 2002). The $(\mu + \lambda)$ environmental selection scheme produces λ offspring from μ parents and these $\mu + \lambda$ individuals compete to form the next generation. This competition is

usually deterministic, with the fittest individuals going forward to the next generation, although variants exist where the selection is performed probabilistically (Bäck 1996). In contrast, the (μ, λ) environmental selection scheme only considers the offspring for inclusion in the next generation. Whilst the $(\mu+\lambda)$ strategy often provides faster convergence, the (μ, λ) reinsertion scheme is generally preferred due to its ability to avoid long periods of stagnation in sub-optimal regions of the search space (Bäck 1996).

2.2.2 Further Concepts in Evolutionary Computation

Exploitation and Exploration

All effective search methods must balance two opposing forces: the *exploitation* of good solutions achieved so far in the search and the *exploration* of new areas of the search space. Too much *exploration* reduces the algorithm to a random search, whilst too much *exploitation* may cause the algorithm to converge prematurely to local optima. Holland (1975) emphasises the importance of achieving a good balance of *exploitation* and *exploration* when using genetic algorithms to adaptively search an unknown space for optimal solutions.

Traditional EA theory views selection as an *exploitative* operator and variation as *explorative* (Eiben and Schippers 1998). However, it can be argued that recombination is also somewhat *exploitative* as it aims to progress the evolutionary search by *exploiting* genetic material already present in the current generation. Studies suggest that decreasing the rate of variation as the search progresses results in improved convergence (Bäck 1992), thus leading to the idea that *exploration* should gradually change into *exploitation* as the search progresses. It has also been shown that recombination operators implicitly reduce their rate of exploration as the population converges (Eiben and Schippers 1998).

Niche Formation in Evolutionary Algorithms

De Jong (1975) has shown that, in multi-modal fitness landscapes, a finite population under going repeated stochastic selection will converge towards a single point in the search space. This phenomenon is termed *genetic drift* and is a well studied problem in population genetics (Hartl and Clark 1997). These stochastic effects can be somewhat reduced by increasing the population size or using a higher mutation rate (De Jong 1975), but incorporating an explicit mechanism to enhance diversity in the population such as *crowding* (De Jong 1975) or *fitness sharing* (Goldberg and Richardson 1987) is more effective in combating genetic drift.

These methods aim to maintain multiple *niches* in the population by modelling competition amongst individuals in the same niche for limited resources. This results in a selection pressure towards less crowded areas of the search space. In the *crowding factor* approach to niche formation (De Jong 1975), the solution from a sample of the parent population, $P(gen-1)$, which is most similar to the child solution is replaced in the current generation, $P(gen)$, at the environmental selection stage. This crowding factor approach has empirically been shown to be weak in multi-modal function optimisation (Deb and Goldberg 1989).

Goldberg and Richardson (1987) introduced a *fitness sharing* scheme whereby individuals in the same niche have their fitness degraded according to the number of similar individuals. Goldberg and Richardson (1987) suggested a general class of sharing functions, but the most commonly used function is of the form,

$$sh(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{share}}\right)^{\alpha} & \text{if } d < \sigma_{share}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

where d is the distance between two solutions in the population, σ_{share} is the *niche size* parameter, and α is used to regulate the shape of the sharing function.

The modified fitness of an individual is equal to its original fitness divided

by its *niche count* (the sum of the sharing values with respect to every member of the population, including itself). This gives us Equation 2.2 below, where f_i' is the modified fitness of the ith individual, f_i is its original fitness, N is the population size, and $d(i,j)$ is the distance between solution i and solution j.

$$f_i' = \frac{f_i}{\sum_{j=0}^{N} sh(d(i,j))} \tag{2.2}$$

The success of fitness sharing relies heavily on the value chosen for the niche size parameter, σ_{share}. Ideally this value would be related to the basin of attraction of the optimum (Deb 2001), however this information is unlikely to be available for a non-trivial problem. Deb and Goldberg (Deb and Goldberg 1989) have proposed methods for estimating this parameter, but they require information about the number of local optima which is generally unknown.

Constraint Handling

> "Virtually all decision making situations involve constraints. What distinguishes various types of problems is the form of these constraints. Depending on how the problem is visualized, they can arise as rules, data dependencies, algebraic expressions, or other forms" (Dhar and Ranganathan 1990).

Problem solving in the real-world almost always involves addressing constraints, whether these constraints are simply bounds on the decision variables or a complex function defining unreachable areas of the search space, the final solution must be *feasible*[1]. However, the basic evolutionary algorithm (as shown in Figure 2.1) is an unconstrained optimisation technique. It is therefore necessary to incorporate *constraint-handling* techniques into the algorithm. A brief summary of the most common methods for handling constraints is given below, however

[1] A solution that satisfies all the constraints on the problem is said to be *feasible*, if a solution violates one or more of the constraints it is said to be *infeasible*.

for a more thorough review the reader is directed to Coello-Coello (2002) or Michalewicz and Schoenauer (1996).

- **Penalty Functions.** The use of penalty functions in solving constrained optimisation problems was introduced in the 1940s (Courant 1943), and is well established in classical optimisation literature (Fletcher 1981). It has also proved popular with the evolutionary computation community (Coello-Coello 2002) due mainly to its simplicity. This approach transforms a constrained optimisation problem into an unconstrained one by modifying the fitness values of infeasible solutions, usually by a function of their constraint violation, to guide the optimiser towards feasible regions of the search space. The main drawback in the use of penalty functions is the difficulty in setting appropriate penalty weightings, although attempts have been made to make these weighting factors adaptive (Hadj-Alouane and Bean 1997).

- **Special Representations and Operators.** Another method of dealing with constraints is to design the representation scheme and variation operators so as to only produce feasible solutions. In practice this is often extremely difficult, and is mainly used in problems where it is difficult to find even one feasible solution. Davis (1991) contains several examples of the use of special representations and operators to tackle complex problems. The main problem with this approach, aside from the potential difficulty in designing the representation and operators, is that an algorithm developed for a specific problem may not generalise well to other problems.

- **Repairing Infeasible Solutions.** Algorithms that repair infeasible solutions have been shown to perform well (Liepins and Potter 1991), especially in combinatorial optimisation problems where it is relatively easy

to repair individuals (Coello-Coello 2002). A typical approach to designing a repair algorithm would be to use a greedy algorithm[2] to find a feasible solution from the infeasible one. These repair algorithms can be a good choice if the cost of repairing a solution is low. However, this approach is problem dependent, with a specific repair algorithm needed for each problem, and so it also does not generalise well.

- **Separating Constraints and Objectives.** One approach to handling constraints and objectives separately is to view the problem as a multi-objective optimisation problem, and the constraints as additional objectives (Coello-Coello 2000b). Evolutionary multi-objective techniques such as Pareto-optimality and non-dominance (see Section 2.4 later) can then be used to assess the fitness of the candidate solutions.

Hybrid Evolutionary Approaches

Moscato (1989) introduces the term *memetic algorithms*[3] to describe EAs that incorporate local search techniques. Researchers have long since recognised the potential benefits of incorporating problem specific information in evolutionary search (Goldberg 1989). These *hybrid* evolutionary algorithms can exploit both the global nature of the evolutionary search process, and the convergence properties of local optimisation methods. The most widely used hybridisation scheme is that of *sequential hybridisation* (see Figure 2.6), where the EA finds a set of candidate solutions and then applies a local search technique (for example, hill-climbing) to improve this population. To quote David Goldberg:

> "the genetic algorithm finds the hills and the hill-climber goes and climbs them" (Goldberg 1989).

---

[2]i.e. an algorithm that makes the move with the largest possible gain at each iteration to arrive at a solution.

[3]Richard Dawkins first defined a *meme* as a unit of information that reproduces itself as people exchange ideas (Dawkins 1976)

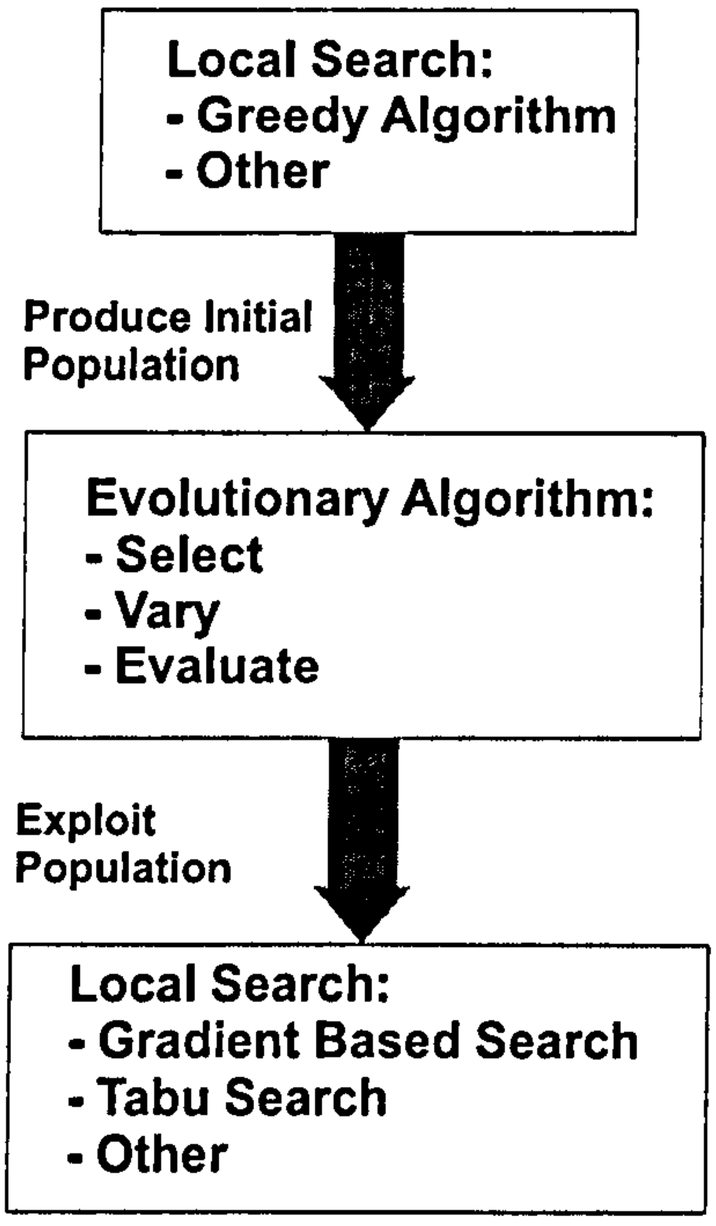


Figure 2.6: Sequential Hybridisation in an Evolutionary Algorithm

Memetic algorithms have been applied to many different problems using countless different local search techniques (see Krasnogor and Smith (2005) or Ong *et al.* (2006) for a more thorough review of memetic algorithms). A major area of application for memetic algorithms is in combinatorial optimisation problems, such as the travelling salesman problem (White and Yen 2004, Moscato and Norman 1992) and the quadratic assignment problem (Merz and Freisleben 1999), where they have been shown to produce good results. The main problem with the incorporation of domain-specific knowledge into an EA is the loss of generality this entails; for every new problem to be solved by the memetic algorithm, new problem specific local search techniques must be developed. Another problem to the memetic algorithm approach is the *ad hoc* nature of the design process (Krasnogor and Smith 2005). There is a lack of theoretical knowledge about appropriate memes to incorporate into the algorithm for a given problem type.

Fitness Approximation in Evolutionary Computation

One of the main difficulties associated with applying population-based optimisation methods (such as evolutionary algorithms) to real-world problems is the large number of objective function evaluations needed to produce a satisfactory solution. In many real-world engineering design problems the computational expense of this evaluation process is prohibitive, due to the complexity of the objective function. One possible solution in these cases is to use *fitness approximation* (see Jin (2005) for an alternative review). Fitness approximation often involves using a simplified model of the problem to calculate the fitness of a candidate solution. Whilst fitness approximation is most commonly used in cases where evaluating the true objective function is very computationally expense (for example, in Varcol and Emmerich (2005) and Regis and Shoemaker (2004)), it can also be used in cases where the fitness function is noisy (Sano and Kita 2000), multi-modal (Liang *et al.* 1999), or ill-defined (Johanson and Poli 1998).

A variety of different meta-modelling techniques exist for developing a simplified model of the objective function, including the use of support vector machines (Abboud and Schoenauer 2002) and Gaussian processes (Ulmer *et al.* 2003). Three of the most widely applied meta-modelling techniques from the literature are outlined below, and the interested reader is directed to Jin *et al.* (2001) or Rasheed *et al.* (2003) for a more thorough discussion and analysis.

- **Polynomial Regression.** Polynomial Regression (PR) models (often known as Response Surface Methodology) have been widely applied to problems in engineering design (see Chen *et al.* (1996) and Simpson *et al.* (1998) for examples). The most common approach is to use a second order polynomial model (see equation 2.3) to represent the objective function, and use either the least squares method or a gradient method to estimate the unknown coefficients (Jin 2005).

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \sum_{j \geq i}^{k} \beta_{ij} x_i x_j \tag{2.3}$$

- **Kriging Methods.** A Kriging model combines both a global regression model of the system, $f(x)$, with a Gaussian random process, $Z(x)$, representing a local deviation from this global model (see equation 2.4 for the general form).

$$y(x) = f(x) + Z(x) \tag{2.4}$$

The Kriging model is extremely flexible due to the wide range of possible correlation functions that can be used in calculating the deviation term (Jin *et al.* 2001). Jin (2005) also notes that a confidence interval of the estimation is easily obtainable; however, for models with a high-dimensionality, the computational expense is significant.

- **Neural Network Estimation.** Artificial Neural Networks (ANNs) are frequently used for function approximation in real-world applications, such as systems identification. Hornik *et al.* (1989) showed that multi-layer feed-forward networks are capable of approximating any continuous function. One problem with developing approximate models using ANNs is the amount of training data needed to obtain a representative model. This training data consists of inputs and outputs from the true objective function and, especially for high-dimensional problems, is often computationally expensive to generate (Jin *et al.* 2000).

Approximate models have been used in many aspects of evolutionary computation. Although they are most commonly used to reduce the computational effort of the evaluation process (see Varcol and Emmerich (2005), Regis and Shoemaker (2004), Bull (1999) and Piccolboni and Mauri (1998) for examples), approximate

models have also been used to inform the initialisation of the population and guide the recombination and mutation operators (Rasheed and Hirsh 2000). Rasheed and Hirsh (2000) observe that using approximate models in this way, rather than to directly evaluate candidate solutions, reduces the risk of misleading the search.

As alluded to above, using approximate models to evaluate candidate solutions can result in a final solution that is sub-optimal. Michalewicz and Fogel (2000) point out that a solution is only a solution with respect to the model used. If the model has a low degree of fidelity, then the final solution found is unlikely to be optimal. To mitigate this problem, Jin *et al.* (2000) suggest using the approximate model alongside the original objective function in the evaluation process. This is termed *controlled evolution.* This strategy aims to prevent the search being misled by evaluating a proportion of the candidate solutions using the true objective function. These *controlled individuals* can then be used to refine the approximate model (Bull 1999).

2.2.3 Advantages and Disadvantages of EC

A key advantage that EAs offer over conventional optimisation techniques is their versatility. EAs have been used to solve many different problems across a diverse range of disciplines (see Section 2.2.4). One of the reasons that EAs are so broadly applicable is that they evaluate candidate solutions based on pay-off information from the objective function, rather than relying on derivative knowledge or other auxiliary information. This allows EAs to be applied in areas where the problem is ill-defined and derivative information is not available. EAs have even been applied in areas where the evaluation of candidate solutions is subjective and is performed by human response to those solutions, such as the design of computer graphics (Sims 1991). EAs have also been shown to be robust when optimising problems with multi-modal, noisy or dynamic fitness landscapes (Holland 1975, Michalewicz and Fogel 2000). This makes them particularly well suited for solving

real-world problems which often exhibit the above characteristics.

This versatility comes at a price, however. Whilst EAs are applicable to a broad range of problems, for any given problem there is likely to be a domain-specific problem solver that will exhibit superior performance. For example, in a resource allocation problem with a linear cost-function and linear constraints, the Simplex method (Dantzig 1963) offers an efficient technique that will out perform evolutionary methods. If, however, the problem has a non-linear cost-function or non-linear constraints (often more realistic in the real-world) then the Simplex method will fail because it assumes that the problem is linear. Another drawback to evolutionary computation is its time complexity. Because EAs iteratively search a population of points, they perform many function evaluations to produce a final solution. For non-trivial objective functions this can be very computationally intensive. One way to address this concern is by using fitness approximation techniques (see Section 2.2.2), whilst another potential solution lies with the development of parallel EA implementations (see Chapter 5).

2.2.4 Applications of Evolutionary Computation

Evolutionary Computation has been applied in many different disciplines to solve a diverse range of problems. This Section aims to provide a brief overview of some applications of interest. For a thorough survey of current EC applications the reader is directed to any of the recent conferences on evolutionary computation. For example, the Genetic and Evolutionary Computation Conference (GECCO) 2005 (Beyer *et al.* 2005) or the IEEE Congress on Evolutionary Computation (CEC) 2005 (IEE 2005).

- **Automatic Programming.** Evolutionary algorithms have been used to evolve computer programs for specific tasks. One commonly used method of doing this is Genetic Programming (Koza 1992). Genetic Programming uses genetic operators to evolve LISP expressions in a

computer program. Genetic Programming has produced *human competitive* results (e.g. the result duplicates a previously patented invention or produces a new patentable invention) in fields ranging from the design of a program for a soccer playing robot (which won it's first two games at RoboCup'97) (Luke 1998) to synthesising the topology for a PID controller (Koza *et al.* 1999).

- **Robotics and Control Systems Engineering.** Evolutionary computation has been applied to many aspects of robotics and control engineering. Evolutionary algorithms have been used to generate optimal trajectories for robot arm movement (Tzafestas *et al.* 1999) and to improve the dexterity of robot manipulators (Erkmen *et al.* 2000). EAs have also been used extensively in tuning controller parameters (Oliveira *et al.* 1991, Porter and Jones 1992, Vlachos *et al.* 1999), as well as in the optimisation of controller structures (Koza *et al.* 2000). Another application of evolutionary computation in control engineering is in system identification (Kristinsson and Dumont 1992), where the EA is used to identify the poles and zeros in the system or to obtain the values of physical parameters.

- **Economics and Social Theory.** EAs have been used to study evolutionary aspects of economics and social theory such as the evolution of cooperation. One example of this is the use of EAs to evolve strategies to the Iterated Prisoner's Dilemma (IPD) (Axelrod 1987). The IPD has been studied extensively in game theory, economics, and political science because it can be seen as an idealised model for real-world phenomena such as arms races (Axelrod 1984). Most of the strategies evolved by the EA were similar to the best human designed strategy (called TIT-FOR-TAT). However, Axelrod (1987) noted that the EA also designed more highly specialised adaptations to specific characteristics of the environment.

- **Scheduling.** Finding good solutions to industrial scheduling problems is of great importance, since both production rates and plant costs are dependent on work schedules. Evolutionary algorithms have had some success in solving the canonical Job-Shop Scheduling Problem (Davis 1985, Mesghouni *et al.* 2004), a problem that is representative of industrial tasks ranging from assembling cars, to scheduling aircraft maintenance. Recent focus in the EC community has been on generating robust and flexible job shop schedules (Jensen 2003). Other scheduling problems solved by EAs include planning maintenance for the (UK) national grid (Langdon 1995) and university course timetabling (Lewis and Paechter 2005).

- **Machine Learning.** Evolutionary computation has been used in many machine learning applications, including classification and prediction tasks such as weather forecasting (Gibson and Burger 2003) and the investigation of protein structures (Schulze-Kremer 1992). The application of evolutionary learning algorithms to pattern recognition problems has yielded promising results (Rizki *et al.* 2002), and evolutionary approaches to knowledge discovery in data mining are becoming increasingly common (Tan *et al.* 2005, Buchtala *et al.* 2005). A common approach in evolutionary machine learning is to use an EA to evolve weights and architectures for a neural network (Tzafestas *et al.* 1999). Chellapilla and Fogel (2001) used this approach to evolve a program that plays checkers to a level that is competitive with human experts.

- **Art and Design.** Evolutionary computation has even been applied in fields where the evaluation of candidate solutions is subjective and has to be performed by human interaction. Sims (1991) used interactive evolutionary computation to evolve 3D plant structures, textures, and animations, whilst Biles (2003) used an interactive EA to learn how to improvise Jazz solos. Interactive evolutionary computation has also been used by Parmee (2002)

to aid a human designer in exploring conceptual design spaces. This approach aims to improve problem definition by extracting optimal design information through interactive evolutionary search.

2.3 Multi-Objective Optimisation

Many real-world problems involve satisfying multiple, often conflicting, objectives. The general form of a multi-objective optimisation problem can be described by an objective vector f and a corresponding set of decision variables x, as shown in equation 2.5. Note that minimisation is assumed through-out this Section with no loss of generality.

$$\min \mathrm{f}(x) = (f_1(x), \ldots, f_n(x)) \tag{2.5}$$

It is unlikely that a single ideal solution to a real-world multi-objective optimisation problem can be found due to conflict between objectives. Instead , the solution to a multi-objective optimisation problem often leads to a set of compromise solutions, where any improvement in one objective leads to a degradation in one or more other objectives. This set of *trade-off* solutions is commonly known as the *Pareto-optimal set*[4].

2.3.1 Pareto-Dominance and Optimality

Optimality in multi-objective problems can be defined in terms of *Pareto-dominance*. The concept of Pareto-dominance (expressed formally in Definition 2.1) allows the comparison of candidate solutions to a multi-objective problem. As an example, consider two candidate solutions, x and y. Solution x can be considered better than solution y if it out performs y in one objective, and

[4]This notion of optimality was originally proposed by Francis Edgeworth in 1881 (Edgeworth 1932) and was later extended by the Italian economist Vilfedo Pareto who used the concept in his studies of economic efficiency and income distribution (Coello-Coello *et al.* 2002).

performs at least as well in the remaining objectives (Figure 2.7 shows this graphically for the two objective case).

Definition 2.1 (Pareto-Dominance) *Given two candidate solutions, x and y, the objective vector* $\mathbf{f}(\mathbf{x})$ *is said to dominate the objective vector* $\mathbf{f}(\mathbf{y})$ *(denoted by* $\mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x})$*) if and only if:*

$$\forall_i \in \{1, \ldots, n\} : f_i(x) \leq f_i(y) \wedge \exists_i \in \{1, \ldots, n\} : f_i(x) < f_i(y)$$

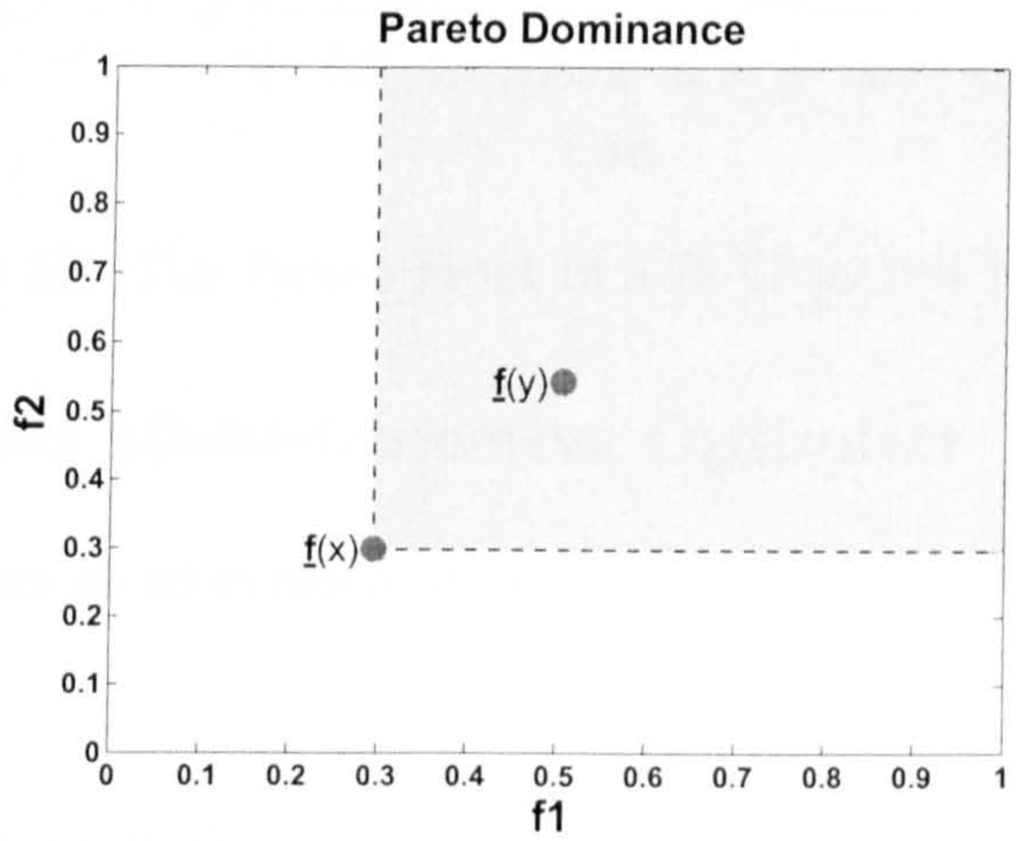


Figure 2.7: Example of Pareto Dominance in Bi-Objective Space

A solution x is said to be *globally non-dominated* or *Pareto-optimal* (see Definition 2.2) if there exists no feasible solution that dominates x. The set of all globally non-dominated solutions is known as the *Pareto-optimal set* and the corresponding set of objective vectors is known as the *Pareto-front* or *trade-off surface* (see Figure 2.8).

Definition 2.2 (Pareto-Optimality) *The candidate solution x is said to be* Pareto-optimal *if and only if:*

$$\neg\exists y : f(y) \prec f(x)$$

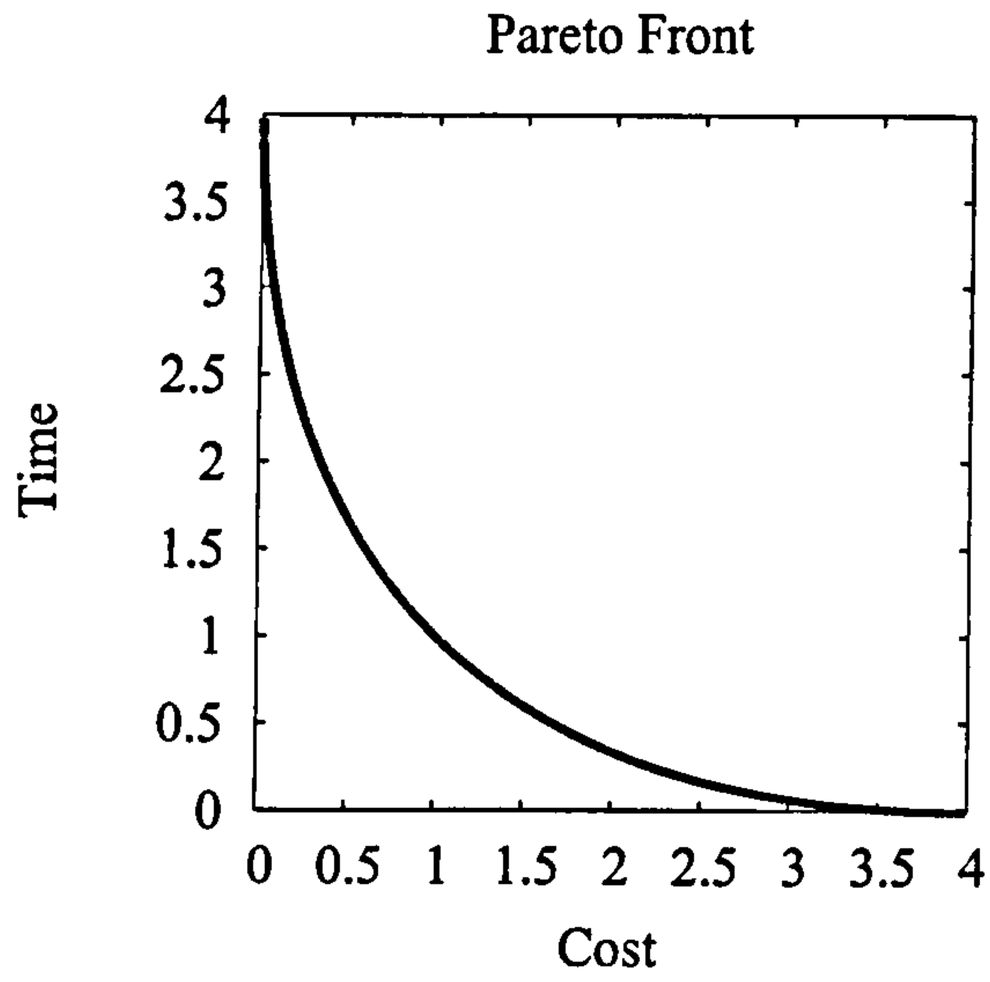


Figure 2.8: The Pareto Front in a Bi-Objective Problem

2.3.2 An Ideal Multi-Objective Optimiser

Deb (2001) describes an ideal multi-objective optimisation procedure (illustrated in Figure 2.9) as:

Step 1: Find multiple trade-off solutions across the entire Pareto-front.

Step 2: Choose one of these trade-off solutions using higher level information.

In step 1 the ideal multi-objective optimiser aims to provide the Decision Maker (DM) with an accurate and diverse representation of the Pareto-front so that, in step 2, a single optimal trade-off solution can be found using the available higher level information. Providing the decision maker with a diverse set of potential solutions to choose from allows the DM to explore their preferences by examining the compromises that exist between objectives. This is important, since the higher level information used by the decision maker to choose a final solution is often subjective - thus the DM's preferences may be altered when they see the potential trade-offs between objectives.

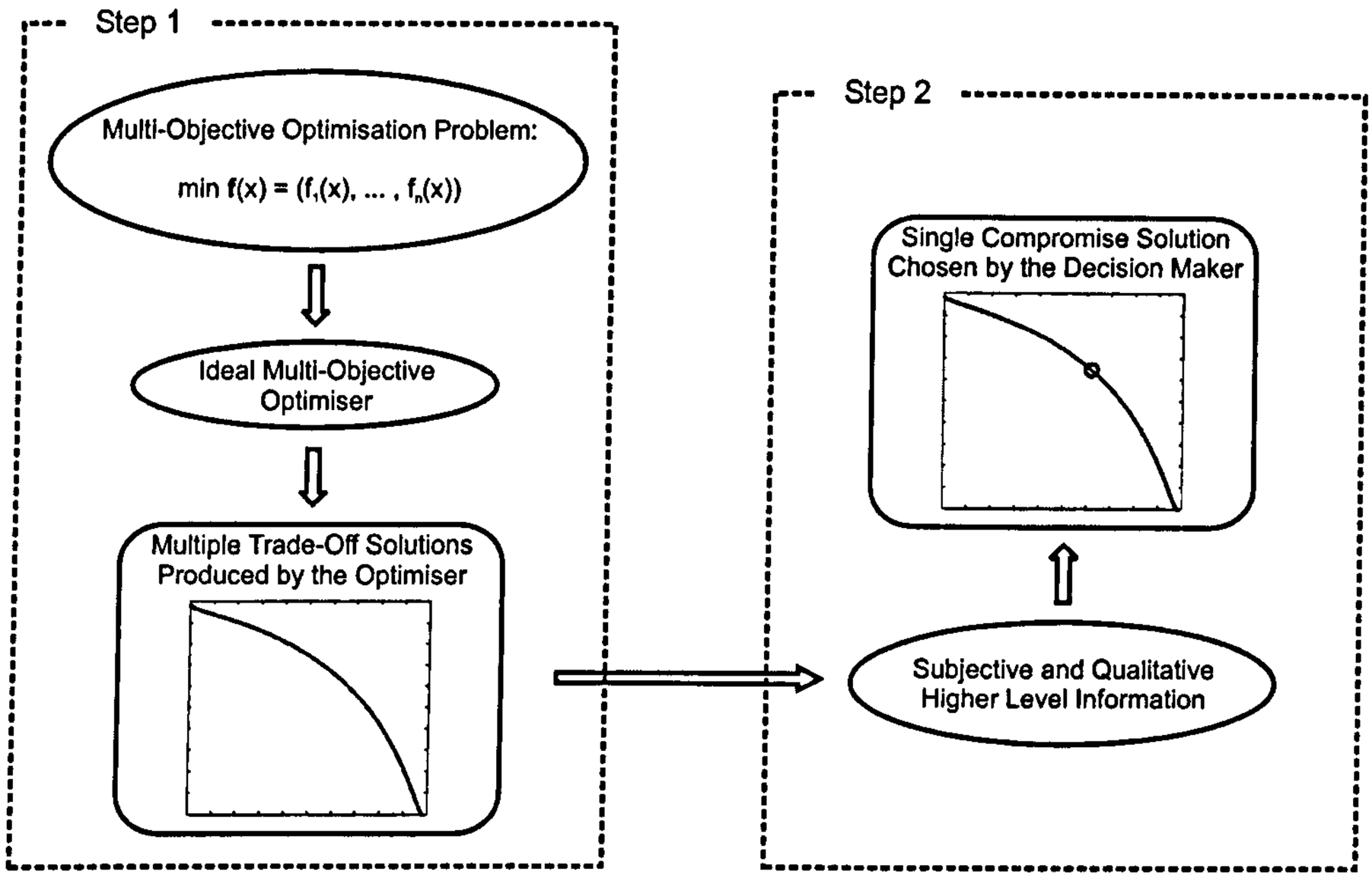


Figure 2.9: An Ideal Multi-Objective Optimisation Procedure

Purshouse (2003) considered three measures of the quality of the solution set[5] produced by an optimiser. These are illustrated graphically in Figure 2.10, and explained below:

- **Proximity.** This is a measure of how close the approximation set is to the true Pareto front. An ideal multi-objective optimiser would produce a set of solutions that correspond to the exact Pareto front. Strategies for the promotion of proximity in an evolutionary multi-objective optimiser are discussed in Section 2.4.3.

- **Diversity.** This is a measure of the distribution of the approximation set, both in the extent and uniformity of that distribution. Good diversity would mean that the set of solutions extends across the entire range of the Pareto front and is uniformly distributed. Strategies for the promotion of diversity in an evolutionary multi-objective optimiser are discussed in Section 2.4.4.

[5]often termed an *approximation set* (Zitzler *et al.* 2003)

- **Pertinency.** This criteria measures the relevance of the approximation set to the decision maker. Ideally the set of solutions should cover the decision maker's region-of-interest (ROI), thus allowing the DM to choose a single compromise solution that satisfies their preferences. See Section 2.4.5 for methods to obtain pertinent solutions in evolutionary multi-objective optimisation.

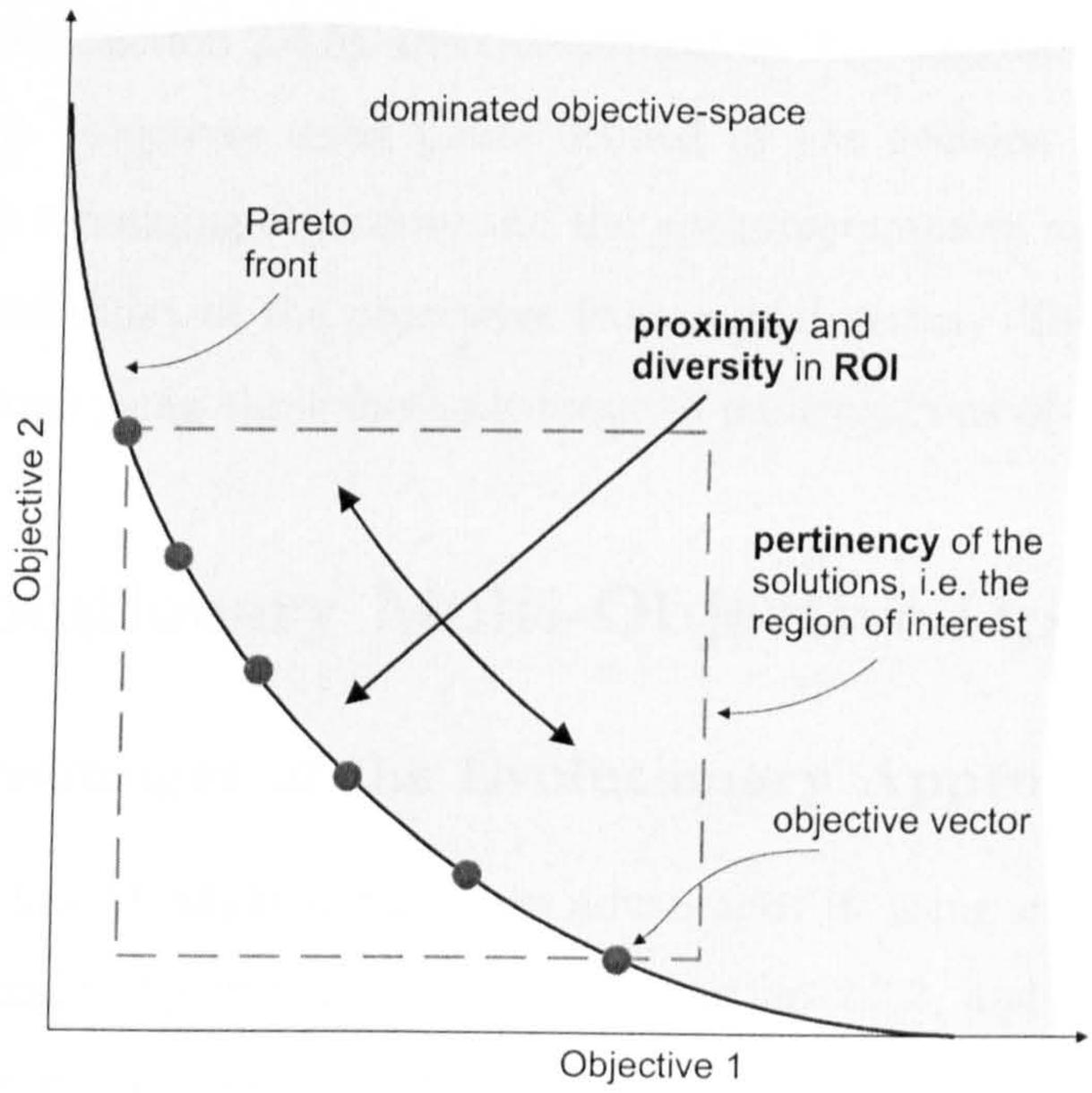


Figure 2.10: The Ideal Solution to a Multi-Objective Problem

2.3.3 Classical Multi-Objective Optimisation Methods

Classical multi-objective optimisation methods usually attempt to convert a multi-objective problem into a single-objective one, and then solve the single-objective problem using conventional optimisation techniques. One of the biggest problems with this approach is that only a single solution from the Pareto optimal set is produced in each run of the optimiser. In the ideal multi-objective

optimisation procedure described in Section 2.3.2, the advantages of presenting the decision maker with multiple trade-off solutions were explained.

There are several techniques that are commonly used to convert a multi-objective problem into a single objective one, but most rely on the decision maker supplying some information about the problem before the optimisation process. For example, the *weighted sum approach* attempts to aggregate multiple objectives using a user supplied set of weights (representing the decision maker's preferences - see Section 2.4.5), the *ϵ-constraint method* suggests constraining all but one of the objectives using limits defined by the decision maker and then optimising the remaining objective, and the *goal programming method* minimises the sum of deviations of the objectives from a goal vector. To obtain a *set* of optimal solutions using these methods requires multiple runs of the optimiser.

2.4 Evolutionary Multi-Objective Optimisation

2.4.1 Advantages of the Evolutionary Approach

This Chapter has highlighted the main advantages in using evolutionary computation for single objective optimisation in Section 2.2.3, and these also apply to optimisation in the multi-objective case. In addition to these, the population based nature of evolutionary algorithms offers a tremendous advantage when dealing with multiple objectives. Step 1 of the ideal multi-objective optimiser in Section 2.3.2 emphasised the importance of finding multiple trade-off solutions across the Pareto-front. Since an EA searches a population of points, it is capable of finding multiple candidate solutions in each generation.

The population-based approach enables a Multi-Objective Evolutionary Algorithm (MOEA) to find an approximation set that perform well with respect to the three measures of solution set quality defined in Section 2.3.2. By using specific operators to maintain diversity in the population (see Section 2.4.4),

the approximation set produced by the MOEA can converge to provide a good distribution over the Pareto front. Also, since an MOEA can optimise multiple objectives simultaneously and present the DM with a family of Pareto-optimal solutions, the decision maker can get an overall perspective of the trade-offs involved in a particular problem. This allows the DM to explore or refine their preferences without re-running the algorithm.

2.4.2 Evolutionary Multi-Objective Optimisation: A History

Early Approaches

The first approaches to Evolutionary Multi-Objective Optimisation (EMO) came in the mid-1980s. Schaffer's Vector Evaluated Genetic Algorithm (VEGA) (Schaffer 1985) is widely credited as the first MOEA, and consists of a simple genetic algorithm with a modified selection mechanism to deal with multiple objectives. In each generation equally sized sub-populations were created for each of the objectives using proportional selection, and then crossover and mutation were performed on the whole population. The biggest problem with this approach is that the selection mechanism is biased against individuals that perform acceptably across all objectives, but excel in none. This results in good compromise solutions often remaining undiscovered. Fourman (1985) and Kursawe (1991) also proposed early MOEAs that were designed to obtain approximation sets, but these suffered from the same major flaw as the VEGA approach.

Other early approaches to EMO drew inspiration from classical optimisation techniques (see Section 2.3.3 for a brief review of classical optimisation methods). These approaches combined multiple objectives using aggregation functions, such as the weighted sum method (Jakob *et al.* 1992) or target vector optimisation (Wienke *et al.* 1992), to form a single objective problem. Whilst methods

that require the aggregation of multiple objectives are conceptually simple and computationally efficient, setting the required weight or target vectors can be difficult for a decision maker. These methods also only generate a single optimal solution in each run, so to generate a trade-off surface requires multiple runs with different weights.

Pareto-Based Approaches

The idea of Pareto dominance (see Section 2.3.1) based fitness assignment was originally proposed by Goldberg (1989) to overcome the problems with the VEGA approach. Goldberg suggested ranking the population on the basis of non-domination, with the globally non-dominated individuals being assigned the highest rank. In Goldberg's ranking scheme these are then removed from contention, and another set of non-dominated individuals are identified and assigned the next highest rank. This process continues until the entire population is ranked. Goldberg also suggested using a *niching* strategy (see Section 2.2.2) to maintain diversity across the Pareto front.

First Generation of Pareto-based MOEAs

Most of the EMO approaches since Pareto dominance based ranking was proposed by Goldberg have been influenced by this idea. In the early-1990s, the first generation of Pareto-based MOEAs were developed. This initial generation was characterised by the simplicity of the algorithms and a lack of methodology to analyse them with (Coello-Coello 2006). The three most cited algorithms from this period are briefly summarised below:

- **Multiple Objective Genetic Algorithm (MOGA).** Fonseca and Fleming (1993) proposed a slightly different Pareto dominance based ranking scheme to that of Goldberg. In MOGA an individuals rank corresponds to the number of individuals by which it is dominated. All individuals in the

globally non-dominated set are assigned the same rank, whilst dominated individuals are penalised according to the population density of the region of the trade-off surface they belong to. Fonseca and Fleming (1993) also implemented fitness sharing in objective space, and provided guidelines for estimating niche sizes according to the properties of the Pareto front. In a study by Van Veldhuizen (1999a) comparing the three MOEAs listed here, MOGA was found to exhibit the best performance.

- **Niched Pareto Genetic Algorithm (NPGA).** This algorithm was proposed by Horn *et al.* (1994) and uses a tournament selection scheme based on Pareto dominance. Two individuals are randomly chosen, along with a subset of the population for comparison. If one of the individuals is dominated by the comparison set and the other is not, then the non-dominated individual is chosen. If neither or both are dominated, then fitness sharing is used to decide the winner.

- **Non-dominated Sorting Genetic Algorithm (NSGA).** Srinivas and Deb (1994) implemented a more direct version of Goldberg's non-dominated sorting concept (Goldberg 1989). In NSGA the individuals in the population are ranked according to which non-domination front they belong to. Initially all individuals of the same rank are assigned equal fitness. Then, to promote diversity, the fitness of individuals is degraded according to the number of neighbouring solutions. Care is taken to ensure no solution in a worse front is assigned better fitness than a solution in a better front. In this way selective pressure towards the Pareto-optimal front is achieved whilst diversity in the approximation set is preserved.

Second Generation of Pareto-based MOEAs

In the late-1990s EMO researchers began to develop a second generation of Pareto-based MOEAs. This second generation of algorithms primarily emphasises

efficiency, focussing particularly on incorporating *elitism* into the environmental selection process (see Section 2.4.3 for an analysis of elitism in MOEAs). These contemporary MOEAs also incorporated more advanced methods for the estimation of population density and the promotion of diversity (see Section 2.4.4 for a review of contemporary diversity promotion methods). The most representative second generation MOEAs are:

- **Strength Pareto Evolutionary Algorithm (SPEA).** Zitzler and Thiele (1999) are often credited with introducing the concept of elitism in EMO with their Strength Pareto Evolutionary Algorithm. SPEA maintains an external non-dominated set into which is copied all the non-dominated individuals at each generation. Each individual in this external non-dominated set is then assigned a *strength* value proportional to the number of members of the population it dominates (similar to the ranking scheme used in MOGA (Fonseca and Fleming 1993)). Fitness is then assigned to each individual in the population according to the *strengths* of the individuals in the external set that dominate it. This fitness assignment procedure thus prefers individuals close to the Pareto front and attempts to achieve a good distribution of solutions across the trade-off surface.

- **Pareto Archived Evolution Strategy (PAES).** The Pareto Archived Evolution Strategy (Knowles and Corne 2000) was designed as a simple *(1+1)* evolution strategy with a reference archive of previous non-dominated solutions. This reference archive is used to identify the appropriate dominance ranking of the current and candidate solution vectors. PAES uses an adaptive crowding procedure that recursively divides up the objective space. Since it is adaptive, this crowding procedure does not require the critical setting of a niche size parameter. Variations on this adaptive grid have been used in several other MOEAs (e.g. the Pareto Envelope-based Selection Algorithm (PESA) (Corne *et al.* 2000) and Coello

and Pulido (2001)'s Micro Genetic Algorithm).

- **Non-dominated Sorting Genetic Algorithm II (NSGA-II).** This algorithm was proposed by Deb, Pratap, Agarwel and Meyarivan (2002) to overcome some of the problems with NSGA (Srinivas and Deb 1994). Specifically, it uses a much faster non-dominated sorting approach as well as a *crowded comparison operator* for selection. This crowded comparison operator takes into consideration both the *non-domination rank* of a candidate solution and its *crowding distance* (a measure of the density of solutions surrounding a particular individual). NSGA-II does not use an external archive like the other MOEAs discussed in this Section, but instead uses a $(\mu + \lambda)$ selection scheme (see Section 2.2.1) to choose the new population from the combined parent and child populations. NSGA-II is currently considered a benchmark algorithm.

- **Strength Pareto Evolutionary Algorithm 2 (SPEA2).** In response to criticisms of the Strength Pareto Evolutionary Algorithmn (Corne *et al.* 2000, Deb *et al.* 2000), SPEA2 was developed by Zitzler *et al.* (2001). The main differences between SPEA and SPEA2 are:

  - SPEA2 uses a fine-grained fitness assignment strategy which takes both dominating and dominated solutions into account.
  - SPEA2 uses an explicit density estimation technique based on the k-th nearest neighbour method.
  - The archive truncation method in SPEA2 prevents boundary solutions being removed.

Another major area of interest in this second generation of Pareto-based MOEAs has been in developing performance measures and test functions to allow for rigorous quantitative analysis and comparison of algorithms. Zitzler *et al.* (2000) defined three goals of an evolutionary multi-objective optimiser:

- Minimise the distance between the approximation set and the true Pareto front.
- Maximise the diversity of the approximation set across the Pareto front.
- Maximise the number of candidate solutions found that belong to the Pareto-optimal set.

Deb (1999b) proposed a methodology for constructing multi-objective test functions that emphasised incorporating features such as multi-modality and discontinuities that might cause MOEAs to have difficulties in achieving these goals. Zitzler *et al.* (2000) later used this methodology to frame six bi-objective problems that have since been used as a benchmark for testing the performance of MOEAs. Deb's methodology has also been extended to deal with an arbitrary number of objectives and seven scalable test functions were proposed (Deb, Thiele, Laumanns and Zitzler 2002) .

In parallel to this development of test functions, several authors proposed unary performance metrics that attempted to quantify the performance of MOEAs in relation to the above goals (see Van Veldhuizen and Lamont (2000) and Deb *et al.* (2000) for examples). However, Zitzler *et al.* (2003) conclusively proved that there exists no unary measure of quality that is able to show that one approximation set is better than another. Instead, the authors suggested the use of binary performance measures (i.e. metrics that consider two algorithms at a time) to overcome this limitation.

2.4.3 Obtaining Good Proximity

The primary goal in evolutionary multi-objective optimisation is finding an approximation set that has good proximity to the Pareto front. These candidate solutions represent optimal trade-offs between objectives. The early approaches to evolutionary multi-objective optimisation (see Section 2.4.2) were primarily

concerned with guiding the search towards the Pareto front, reflecting the importance of this goal.

Convergence to the Pareto front is mainly driven by mating selection (see Section 2.2.1), where the best candidate solutions are assigned the highest fitness (and thus have the best chance of contributing to the next generation). As can be seen from Section 2.4.2, several techniques have been proposed to solve the problem of assigning scalar fitness values to individuals in the presence of multiple objectives - with Pareto-based methods generally being considered the best. Several variants of Pareto based fitness assignment methods exist (see Section 2.4.2 or Zitzler *et al.* (2004) for more information), but the general procedure is to rank individuals in the approximation set according to some dominance criterion, and then map fitness values to these ranks (often via a linear transformation). Mating selection then proceeds using these fitness values.

The proximity of the approximation set to the Pareto front can be enhanced by the use of *elitism*. Elitism aims to address the problem of losing good solutions during the optimisation process (Zitzler *et al.* 2004), either by maintaining an external population of non-dominated solutions (commonly referred to as an *archive*), or by using a $(\mu + \lambda)$ type environmental selection mechanism (see Section 2.4.2 for examples of the use of elitist strategies in MOEAs). Studies have shown that elitist MOEAs often perform favourably when compared to their non-elitist counterparts (Zitzler and Thiele 1999, Zitzler *et al.* 2000). Elitism has also been shown to be a theoretical requirement to guarantee convergence of an MOEA in the limit (Rudolph and Agapie 2000).

In archive based elitism, the archive can be used either just to store good solutions generated by the MOEA, or can be integrated into the algorithm with individuals from the archive participating in the selection process. Some mechanism is often needed to control the number of non-dominated solutions in the archive, since the archive is usually a finite size and the number of non-dominated individuals can potentially be infinite. Density based measures (see

Section 2.4.4) are commonly used in this archive reduction.

(μ+λ) type environmental selection schemes originated in Evolution Strategies and involve both the parent population and the child population competing against each other for selection (see Section 2.2.1 for a brief summary).

2.4.4 Obtaining Good Diversity

Most MOEAs use density information in the selection process to maintain diversity in the approximation set. However, diversity preservation is usually seen as a secondary consideration after obtaining good proximity to the Pareto front. This is because, as Bosman and Thierens (2003) state:

> "...since the goal is to preserve diversity *along* an approximation set that is as close as possible to the Pareto optimal front, rather than to preserve diversity in general, the exploitation of diversity should not precede the exploitation of proximity."

Goldberg (1989) initially suggested the use of a *niching* strategy in EMO to maintain diversity, with most of the first generation of Pareto based MOEAs using the concept of fitness sharing from single-objective EA theory (Fonseca and Fleming 1993, Horn *et al.* 1994, Srinivas and Deb 1994). Fitness sharing, discussed earlier in Section 2.2.2, was originally proposed by Goldberg and Richardson (1987) to overcome the problem of genetic drift in multi-modal fitness landscapes. In evolutionary multi-objective optimisation, fitness sharing can enable an MOEA to find multiple diverse solutions along the Pareto front.

As mentioned in Section 2.2.2, the success of fitness sharing is dependent on the choice of an appropriate niche size parameter, σ_{share}. Whilst several authors proposed guidelines for choosing σ_{share} (Deb and Goldberg 1989, Fonseca and Fleming 1993), Fonseca and Fleming (1995) were the first to note the similarity between fitness sharing and kernel density estimation in statistics.

This provided the EMO community with a set of established techniques for automatically selecting the niche size parameter, such as the *Epanechnikov estimator* (Silverman 1986).

Many of the second generation of Pareto based MOEAs include advanced methods of estimating the population density, inspired by statistical density estimation techniques (see Section 2.4.2 for a brief overview of density estimation methods used in MOEAs). These can be mainly classified into *Histogram techniques* (such as that used in PAES (Knowles and Corne 2000)) or *Nearest Neighbour density estimators* (such as that used in SPEA2 (Zitzler *et al.* 2001) and NSGA-II (Deb, Pratap, Agarwel and Meyarivan 2002)). These estimates of population density can be used in both mating selection and environmental selection. In mating selection these density estimates are commonly used to discriminate between individuals of the same rank. Individuals from a less dense part of the population are assigned higher fitness and thus have a higher chance of contributing to the next generation.

Density estimation in environmental selection is commonly used when there exists more locally non-dominated solutions than can be retained in the population. For example, in archive based elitism, density based clustering methods are often used to reduce the archive to the required size. Non-dominated solutions from sparser regions of the search space are again preferred over those from regions with higher population densities. This aims to ensure that the external population contains a diverse set of candidate solutions in close proximity to the Pareto front.

2.4.5 Obtaining Good Pertinency

Solving multi-objective problems generally involves two distinct processes: *search* and *decision making*. Traditionally these two aspects of multi-objective problem solving have been considered independently, with little crossover between the two fields. Evolutionary multi-objective optimisation provides a powerful tool in

searching for multiple trade-off solutions, but a single solution still needs to be chosen by the decision maker. To assist the DM in choosing a single compromise solution from the Pareto optimal set, it is useful to be able to present only *pertinent* solutions (i.e. solutions within the region-of-interest of the decision maker) for consideration. This ROI can be defined in terms of the decision maker's preferences.

Horn (1997) classifies Multi-Criteria Decision Making (MCDM) in EMO according to when the preference information is used in the multi-objective problem solving process:

1. The preference information is incorporated before the search (preferences $\Rightarrow$ search). This is referred to as *a priori* preference articulation.

2. The preference information is used after the search to choose from the approximation produced by the optimiser (search $\Rightarrow$ preferences). This is referred to as *a posteriori* preference articulation.

3. The preferences of the DM are incorporated during the search (preferences $\Leftrightarrow$ search). This is referred to as *progressive* preference articulation, and allows the decision maker to alter their preferences throughout the search.

The use of preference articulation in solving multi-objective problems has been overlooked somewhat in the EMO literature. The following sections offer a review of preference articulation and its application in evolutionary multi-objective optimisation. Coello-Coello (2000a) offers an alternative survey, with an emphasis on the Operations Research (OR) perspective.

A Priori Preference Articulation

The most common method of handling multiple criteria in classical optimisation is to use the DM's preferences *a priori* to aggregate multiple objectives into a single objective (see Section 2.3.3 for a brief review of classical multi-objective

optimisation). Many early approaches to EMO also used this idea (see Section 2.4.2) to overcome the difficulties associated with optimising multiple objectives. A major drawback with this aggregation approach is that only a single trade-off solution is generated for each run of the algorithm.

A priori preference articulation can also be incorporated into Pareto based MOEAs to produce multiple trade-off solutions in the decision maker's region-of-interest. Cvetkovic and Parmee (1999) proposed a method for transforming qualitative preferences into quantitative weights using a fuzzy pairwise comparison of objectives. These weights can then be used in either a weighted sum aggregation of objectives, or in a modified Pareto based MOEA (Parmee *et al.* 2000). In the modified Pareto based MOEA, the weights are used to bias the dominance relationship towards the regions of the search space preferred by the DM. It is worth noting that, for problems with many objectives, the potential number of pairwise comparisons needed in the fuzzy preference articulation stage is very large (Cvetkovic and Parmee 2002), though transitivity and other preference properties usually reduce that number.

Branke *et al.* (2001) proposed a Guided Multi-Objective Evolutionary Algorithm (G-MOEA) that also uses weights to modify the definition of dominance. In G-MOEA these weights represent maximally acceptable trade-offs between objectives (specified by the DM). For example, the DM could specify that one unit of f_1 is worth at most a units of f_2. This is then used to modify the definition of dominance so that each candidate solution dominates a larger region of objective space. However, for problems with many criteria to consider, specifying maximal trade-offs between pairs of objectives may be difficult for a DM. This method of *a priori* preference articulation also does not work well on non-convex regions of the search space.

Multi-criteria decision making and decision aid approaches that have been developed in the operations research community can also be integrated into evolutionary multi-objective optimisation schemes. Rekiek *et al.* (2000) used the

PROMETHEE-II (Preference Ranking Organisation METHod for Enrichment Evaluations (Brans *et al.* 1986)) outranking approach to order the candidate solutions produced by an evolutionary algorithm. This method required the DM to decide on the weights to be used *a priori.*

Greenwood *et al.* (1996) proposed using elements of *Imprecisely Specified Multi-Attribute Value Theory (ISMAUT)* to perform imprecise ranking of attributes (White III *et al.* 1984). Instead of *explicitly* ranking the objectives, the decision maker is asked to rank a small number of candidate solutions. This ranking of candidate solutions thus *implicitly* defines a ranking of the objectives. This preference information is then incorporated *a priori* into the fitness assignment procedure used by the MOEA. The ISMAUT model used in this preference articulation method assumes that all the objectives in the problem are mutually preferentially independent. Whilst this assumption holds for many problems, if the attributes are mutually preferentially dependent, a more complex value function would be needed.

Deb (1999a) proposed using the decision maker's preferences to introduce bias into the fitness sharing strategy of an MOEA. As mentioned early in Section 2.4.4, fitness sharing aims to obtain a good distribution of solutions across the entire Pareto front. By using normalised weights (representing the DM's preferences) to modify the distance metric used for fitness sharing, the MOEA is able to find a biased distribution of solutions. This biased sharing approach produces a higher density of solutions towards the decision maker's preferred objective. However, as Deb et al. (Deb *et al.* 2006) point out, the approach can not be used to obtain a biased distribution anywhere on the Pareto front in a controlled manner. Branke and Deb (2004) built on the idea of using biased sharing introduced by Deb (1999a), and proposed a more flexible *biased crowding distance measure.* This approach modified the *crowding distance metric* used to measure population density in NSGA-II (Deb, Pratap, Agarwel and Meyarivan 2002) to provide a much more flexible preference articulation strategy that allows better control

over the decision maker's ROI.

Deb *et al.* (2006) proposed using concepts from the reference point approach (Wierzbicki 1980) with an MOEA to attempt to find a set of preferred Pareto optimal solutions near the DM's ROI. In the proposed scheme the decision maker can specify multiple reference points and the MOEA will generate a set of candidate solutions in the neighbourhood of each of these points, unlike in classical methods where only a single reference point can be supplied. The main ideas behind this methodology are that:

1. Solutions close to the reference points are emphasised.

2. Solutions within the close neighbourhood of a near reference point solution are de-emphasised so as to promote diversity around the reference point.

Deb *et al.* (2006) have shown that this method is indifferent to the shape of the Pareto front and is applicable to problems with many objectives. The reference points in this method are supplied by the DM *a priori.*

The main disadvantage with *a priori* preference articulation methods is the need for the decision maker to express his or her preferences before performing the search. In many real-world problems the decision maker may not be sure of their preferences prior to the optimisation process. Preferences are often informed by subjective and qualitative higher level information so, without a thorough understanding of the problem, it may be difficult to define a set of exact preferences *a priori.* In some cases the preferences of the DM may even change during the search (for example, if they see that large gains can be achieved in one objective with only small degradations in another objective). *A priori* preference articulation methods do not offer the decision maker an overall perspective of the available trade-offs in a problem. Ideally, the DM would be able to explore his or her preferences by examining the compromises that can be made between objectives.

A Posteriori Preference Articulation

A posteriori preference articulation is assumed in most EMO approaches. The traditional approach to EMO has been to present the DM with a diverse approximation set with good proximity to the true Pareto front. It is then the job of the DM to use any higher level information available to choose a single compromise solution. Deb (2001) suggests three post-optimal techniques to help the decision maker find a single trade-off solution:

1. *Compromise Programming* (Yu 1973) can be used to find the candidate solution which is closest to some ideal reference point in objective space.

2. The solution with the maximum *marginal rate of substitution* (Miettinen 1999) can be chosen. The marginal rate of substitution is the amount of improvement that can be made in one objective by sacrificing a unit decrement in another objective.

3. *Pseudo-weight vectors* can be calculated for each solution in the approximation set. The DM can then choose the solution that has the closest weight vector to their preferences.

Another *a posteriori* approach is to use MCDM methods from the operations research literature. Massebeuf *et al.* (1999) used the PROMETHEE-II (Brans *et al.* 1986) outranking approach to choose a single compromise solution from the Pareto-optimal set of solutions produced by an MOEA (based on the preferences expressed by a DM).

A posteriori preference articulation methods allow the DM to get an overall perspective of the possible compromises that can be made between objectives, thus overcoming the main disadvantage associated with *a priori* methods. In *a posteriori* preference articulation, the DM is offered a better understanding of the objective space of a multi-objective problem. By choosing a single trade-off solution from the Pareto-optimal set, the potentially subjective and qualitative

nature of the higher level information available is less important because the decision maker can explore the possible compromises available. This potentially allows the DM to make a better decision.

The main disadvantages to *a posteriori* preference articulation are the complexity of the post-optimal search and the (potentially) large amount of irrelevant solutions presented to the DM. Psychological studies have shown that humans cannot effectively deal with large amounts of data (Miller 1956), and thus choosing a single solution from a diverse approximation set may prove difficult for a DM.

Progressive Preference Articulation

Interactive approaches to MCDM (i.e. the *progressive* articulation of preferences) are often preferred by the OR community over *a priori* and *a posteriori* methods (Cohon and Marks 1975). *Progressive* preference articulation is less common in the evolutionary multi-objective field however.

Fonseca and Fleming (1993) were the first to recognise the need to allow the decision maker to *progressively* refine their requirements during the run of an MOEA. They proposed modifying the fitness assignment operator used in MOGA (see Section 2.4.2) to utilise goal information supplied interactively by the DM. These goals were then used in a similar way to the conventional goal attainment method (Aouni and Kettani 2001) to focus the search on the ROI defined by the DM. Fonseca and Fleming (1998) later proposed an extension to their goal based preference articulation method (Fonseca and Fleming 1993) that incorporated both goal and priority information into a transitive relational operator, termed *preferability*. This *preferability* operator allows the DM to set goal and priority levels for each of the objectives at any point in the optimisation process. This information is then used in a modified definition of dominance which encompasses Pareto-optimality, the lexicographic method, constrained optimisation, constraint satisfaction, and goal programming.

Focussing the optimisation process on areas of the search space that are likely to produce compromise solutions of interest to the DM can potentially reduce the computational effort of the search, because the optimiser is no longer concerned with providing a global description of the trade-off surface. Instead, the goal of the optimiser is to find candidate solutions in a sub-region of the Pareto front that the decision maker is interested in. Whilst, in many real-world problems, it may be difficult for the DM to provide accurate preference information *a priori*, it is likely that as the search progresses the decision maker will gain a greater understanding of the trade-offs involved between objectives. This may allow the decision maker to make better informed decisions about their aspiration levels (in terms of goals and priorities) as more information about the problem is gathered. Allowing the decision maker to interactively refine their preferences as the search progresses can thus improve the quality of the solutions produced by the optimiser, as well as improving the efficiency of the optimisation process.

The main disadvantage with *progressive* preference articulation strategies is the effort required from the decision maker. In some cases, the DM may be required to interact with the optimisation process at every generation (Takagi 2001). To overcome this need for continual interaction between the optimiser and the DM, Fonseca and Fleming (1993) proposed the use of an automated DM such as an expert system. Todd and Sen (1999) implemented such a system using an Artificial Neural Network (ANN) to model the preferences of the decision maker. In this system, the decision maker is asked to give scores to a small set of candidate solutions chosen from the current generation. These scores are then used to train the neural network using back-propagation. This *preferencing* process takes place after every 10 generations, with the set of candidate solutions being chosen to provide a broad cross Section of preferences.

2.4.6 Applications of Evolutionary Multi-Objective Optimisation

Although interest in using evolutionary multi-objective optimisation to solve real-world problems has increased considerably since the late-1990s, much of the focus in the literature is on solving problems with a small number of objectives (see Abido (2006), Anderson *et al.* (2005), Duda and Osyczka (2005), and Lahanas *et al.* (2003) for examples). Coello-Coello *et al.* (2002) showed that the majority of applications of MOEAs only consider bi-objective problems. This is probably due to difficulties in understanding and visualising problems with many objectives[6], as well as the strong theoretical focus on bi-objective problems. However, in many real-world problems it may be necessary to consider more than two or three objectives to obtain a satisfactory solution. This Section will introduce some applications of interest, where evolutionary multi-objective optimisation has been used to solve problems with many objectives. The interested reader is directed to recent conferences in EC or EMO (for example, the International Conference on Evolutionary Multi-Criterion Optimisation (EMO) 2005 (Coello *et al.* 2005)) for a thorough survey of current applications of evolutionary multi-objective optimisation.

- **Automotive Engineering.** Laumanns *et al.* (2001) used evolutionary multi-objective optimisation to perform a design space exploration of the road train concept. This problem had 10 objective functions designed to provide a complete characterisation of vehicle performance. Evaluation of candidate solutions produced by the optimisation algorithm was performed by simulation, and the resulting design space exploration produced two significant results. Firstly, the exploration showed that road trains consume considerably less fuel per load than standard trucks, and secondly, the use

[6]The phrase *many objective* has been suggested in the OR community to refer to problems with more than the standard two or three objectives (Farina and Amato 2002).

of more powerful engines in the road trains resulted in much better climbing ability without an excessive increase in fuel consumption.

Benedetti *et al.* (2006) have used evolutionary multi-objective optimisation in the design of a racing car tyre-suspension system. This problem had 24 objective functions representing different performance indexes; however, after correlation analysis it was shown that these 24 objective functions could be reduced to 13 for the purpose of optimisation. The NSGA-II multi-objective evolutionary algorithm (Deb, Pratap, Agarwel and Meyarivan 2002) was used to generate a set of nondominated solutions and then a fuzzy optimality operator (Farina and Amato 2004) was used to perform *a posteriori* selection from this nondominated set. The approach proposed by Benedetti *et al.* (2006) gives a design configuration that exhibited better performance than the reference car.

- **Telecommunications Engineering.** Manos *et al.* (2005) have used multi-objective evolutionary algorithms in the design of two different photonic devices for use in telecommunications. The first application of EMO was to a 4 objective problem in the design of Fibre Bragg Gratings (FBGs). FBGs are used in optical fibre communication systems to combine, divide and filter digital light signals. NSGA-II (Deb, Pratap, Agarwel and Meyarivan 2002) was used to identify two distinct clusters of nondominated solutions representing two different types of FBG. Using domain specific knowledge, a design was selected from the second cluster as solutions in this cluster resulted in better signal quality. Manos *et al.* (2005) also used NSGA-II (Deb, Pratap, Agarwel and Meyarivan 2002) in the bi-objective design exploration of Microstructured Polymer Optical Fibres (MPOFs). NSGA-II was able to identify a set of optimal trade-off solutions between the parabolic average index profile (which corresponds to the bandwidth of the fibre) and the field hole overlap (which corresponds to transmission loss

characteristics).

- **Finance.** Schlottmann *et al.* (2005) have applied evolutionary multi-objective optimisation to financial risk management. By using EMO methods, the need to aggregate noncommensurable risk metrics is avoided. Four objectives are considered; expected rate of return, market risk, credit risk, and required capital for operational risk compensation. The optimisation of investment portfolios was performed using NSGA-II (Deb, Pratap, Agarwel and Meyarivan 2002) and an example using historical market data in the calculation of the objectives was presented. The key advantages to the integrated risk management strategy proposed by Schlottmann *et al.* (2005) are, firstly, that aggregation of incompatible risk figures is not needed, and secondly, that the risk manager is provided with a number of optimal solutions which can be used in an analysis of the trade-offs between different risk types and the expected rate of return.

2.5 Summary

Decision making in engineering design can often be aided by using evolutionary computation to solve optimisation problems. Many real-world problems involve the satisfaction of multiple, conflicting objectives and, as this Chapter has shown, evolutionary algorithms can provide a powerful tool for tackling this kind of problem. The main drawback to an evolutionary approach to solving engineering design problems is the computational complexity of the optimisation process.

This Chapter has aimed to provide a thorough overview of evolutionary multi-objective optimisation. In Section 2.2 evolutionary computation techniques were introduced and, in Section 2.2.1, a generic evolutionary algorithm was outlined. More advanced concepts in evolutionary computation were discussed in Section 2.2.2, and, in Section 2.2.3, the advantages and disadvantages of evolutionary

computation were examined. Section 2.2.4 outlined some of the diverse range of problems that evolutionary computation has been applied to.

In Section 2.3 the key concepts from multi-objective optimisation were introduced, notably the concept of Pareto-dominance in Section 2.3.1, and the requirements of an ideal multi-objective optimiser were discussed (see Section 2.3.2). A brief review of classical multi-objective optimisation methods was given in Section 2.3.3. In Section 2.4.1 the suitability of the evolutionary approach to multi-objective optimisation was examined, whilst in Section 2.4.2 a historical overview of the evolutionary multi-objective optimisation field was presented. Sections 2.4.3, 2.4.4, and 2.4.5 examined methods for achieving a high quality approximation set with respect to the three measures of solution set quality discussed in Section 2.3.2.

The concepts introduced in this Chapter will be built on in Chapters 5 and 6 in an attempt to mitigate the computational complexity of evolutionary optimisation methods. Chapter 5 will take advantage of the population based nature of evolutionary optimisation to investigate the parallelisation of evolutionary algorithms - especially in large scale distributed environments such as computational Grids (see Chapter 3 for a review of Grid computing). The approach taken in Chapter 6 is to reduce the amount of wasted effort in the optimisation process by tightly integrating search and decision making to allow a user to computationally steer the evolutionary optimisation process.

Chapter 3

Review of Grid Computing

3.1 Introduction

Modern science and engineering is increasingly concerned with complex simulation and data analysis. 10 years ago biologists were attempting to sequence the genomes of bacteria, now researchers have succeeded in sequencing the entire human genome (The International Human Genome Sequencing Consortium 2004). As mentioned in Chapter 2 earlier, optimisation in real-world engineering design problems is also computationally expensive, often requiring complex computer models to be run many thousands of times. Several particle physics projects, like the Large Hadron Collider (LHC) at CERN, are projected to require larger amounts of computational resources than are currently available, as well as the ability to collaborate with researchers all over the world. When the LHC comes on-line in 2007 it will produce approximately 15 petabytes (PB) of data per year - roughly a million times the storage capacity of an average desktop computer. To perform even basic data analysis on this amount of data is likely to require at least 20 teraflops (TFLOPS) of computing power, whilst more detailed analysis will require orders of magnitude more power (Foster 2000) (currently the fastest supercomputer in the *Top500 List* (2006) is Blue Gene/L which can manage 280

TFLOPS). This need for greater computing power has led to many advances in scalable computing, such as distributed peer-to-peer (P2P) networks for problem solving (e.g. the United Devices Human Proteome Folding Project (United Devices 2004) aimed at predicting protein structures based on human genome sequence data).

Computational grids are an extension to this concept of scalable computing. They are internet-based networks of geographically distributed computing resources that scientists can share, select from, and aggregate to solve large scale problems (Chetty and Buyya 2002). One of the central ideas behind Grid computing is to make computing resources available like electricity (hence the name). Grid technologies seek to enable scientists not only to share computing power, but to link data, computers, sensors, and other resources into a virtual laboratory (Foster 2000). They aim to do this by (Chetty and Buyya 2002):

- Enabling resource[1] sharing.
- Providing transparent access to remote resources.
- Allowing on-demand aggregation of compute resources across multiple, geographically distributed locations.
- Providing remote access to databases and software.
- Reducing execution times for large-scale data processing applications.

Foster (2000) predicts that, as the technologies mature, it will become common place for communities of scientists to create *Science Grids* linking their resources to support communication, data access, and computation. He predicts the result to be integrated grids in which different types of problems are routed to the

[1]Note that the term *resource* does not refer solely to computers. As mentioned earlier, the Grid offers the potential to share data and data storage facilities, sensors, and even electron microscopes (Lee *et al.* 2003)

appropriate resources (i.e. dedicated supercomputers for specialised problems that require tightly coupled resources, and clusters of idle workstations for more latency tolerant problems such as data analysis).

This Chapter aims to provide an overview of the current state of Grid computing research. It starts, in Section 3.2, with a historical perspective of some important enabling technologies. Section 3.3 then provides a survey of the current technological state of the Grid computing paradigm, including a discussion of some important issues in providing production quality Grid computing environments to scientists and engineers. Some real-world examples of computational Grids being used to enable large scale scientific and engineering projects are then presented in Section 3.4.

3.2 History of Grid Computing

Many of the key concepts underpinning the development of Grid computing are not new. As far back as 1965 the developers of the Multics operating system (Corbató and Vyssotsky 1965) proposed a vision of a "computing facility as a utility like a power company or water company" (Vyssotsky *et al.* 1965). Len Kleinrock, upon the installation of the first ARPANET[2] node in 1969, also suggested:

> "we will probably see the spread of computer 'utilities', which, like present electric and telephone utilities, will serve individual homes and offices across the country."(Klienrock 1969)

Licklider and Taylor (1968) envisioned a distributed computer network, very much like the concept of Grid computing today, where computers would enable

[2]The ARPANET was developed by the Advanced Research Projects Agency (ARPA) of the US department of defence. It was the worlds first packet-switching network, and eventually led to the global network now known as the *Internet*.

collaboration between users. Larry Roberts, the principle architect behind ARPANET, recalled (Segaller 1998):

> "Lick had this concept of the intergalactic network which he believed was [that] everybody could use computers anywhere and get at data anywhere in the world. He didn't envision the number of computers we have today by any means, but he had the same concept - all of the stuff linked together throughout the world, that you can use a remote computer, get data from a remote computer, or use lots of computers in your job."

It is only recently, however, that the combination of technology trends and research advances has made the realisation of these visions possible (Foster 2002a). The rest of this Section will provide a brief history of the development of these enabling technologies (shown as a time-line in Figure 3.1).

3.2.1 Networking History

As mentioned briefly earlier, the first ARPANET nodes were installed in 1969, and, by December of that year, all four nodes (situated at UCLA, Stanford Research Institute, University of California - Santa Barbara, and the University of Utah) were connected. By the mid-1970s there were more than 50 nodes connected to the ARPANET; including universities, government contractors, and military sites. A key development during this time was the Transmission Control Protocol (TCP) (Cerf *et al.* 1974), which later became TCP/IP (see Tannenbaum (1996) for more detailed information about TCP/IP). TCP/IP was designed to provide extremely reliable communications by incorporating powerful error detection and retransmission capabilities. In 1983 the ARPANET switched to using TCP/IP as its communication protocol.

Another key development in networking technology was the invention of the *World Wide Web*. The web was originally proposed by Tim Berners-Lee in

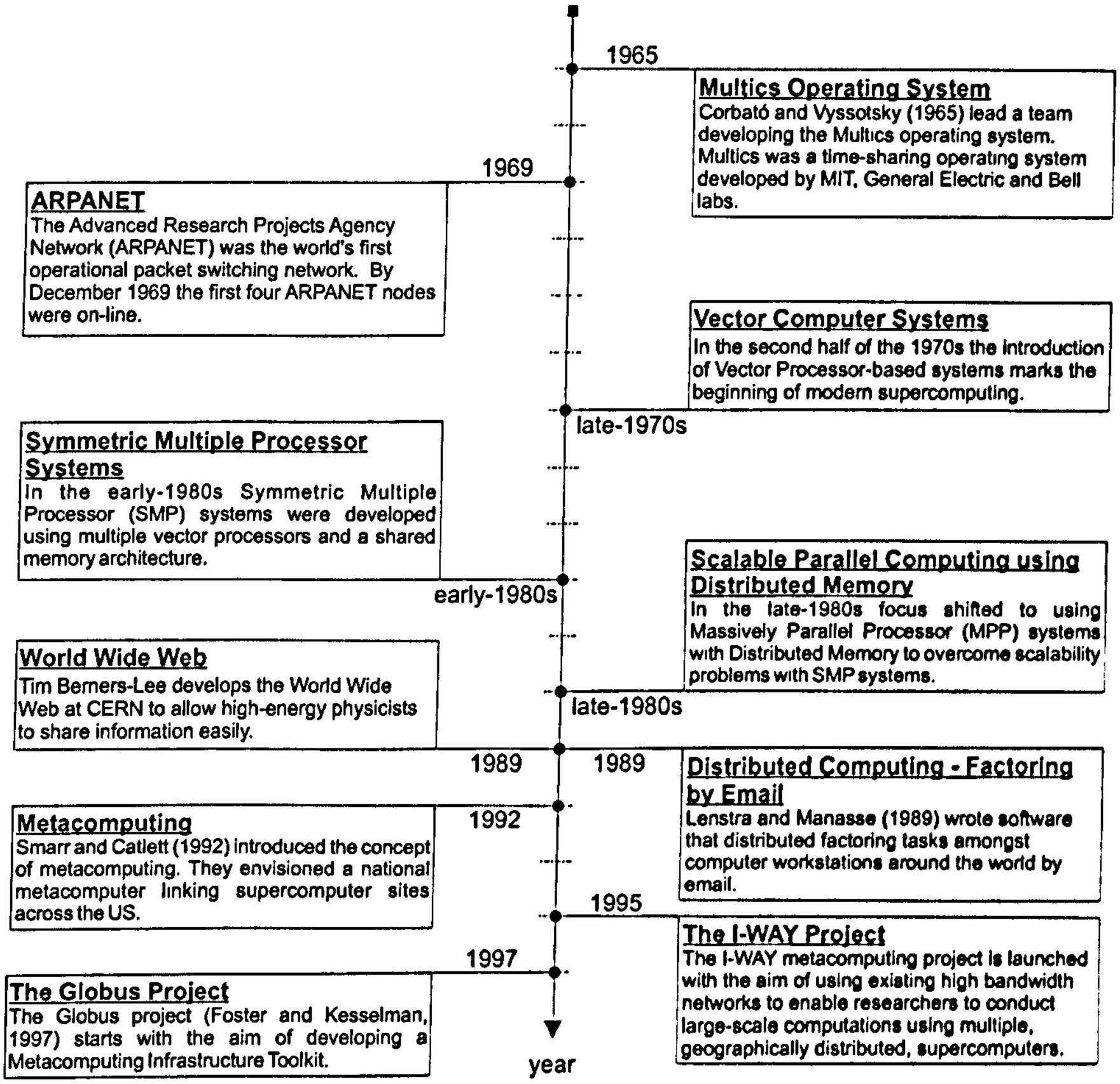


Figure 3.1: A Time-line Charting Key Developments in the History of Grid Computing

1989 (Berners-Lee 1999) as a way of sharing information amongst high-energy physicists at CERN. Initially the proposal did not generate much interest so, in 1990, Berners-Lee built all the tools necessary for a working system: the first web client, the first web server, and the first web pages (describing the project). The web pages were written in HTML (the Hyper Text Mark-up Language) which provided a standard, but loosely structured, way of representing information, whilst HTTP (the Hyper Text Transfer Protocol) combined with the web client and the web server allowed transparent access to this information.

3.2.2 Supercomputing History

Modern supercomputing can trace its origins back to the late-1970s with the introduction of *Vector Processors* (Dongarra 2006). Vector processor-based computer systems are designed to perform arithmetic operations on multiple data elements simultaneously, and offered a performance increase of at least an order of magnitude over conventional systems at the time. Later, in the first half of the 1980s, Symmetric Multiple Processor (SMP) systems that used multiple vector processors running in parallel became common. These systems used a *shared memory* architecture (see Figure 3.2) and had the advantage of being somewhat scalable.

In the late-1980s, the focus in supercomputing shifted towards scalable parallel computing using distributed memory (see Figure 3.3). These new systems aimed to overcome the scalability limits of the shared memory systems (i.e. the rate at which data can be sent to and from memory) and were motivated by an increase in performance and decrease in price of off-the-shelf hardware (Dongarra 2006). In these Massively Parallel Processing (MPP) systems, each node is essentially a computer with its own memory and a link to the other nodes. A continuation of this trend was the launch of the Beowulf Project in 1994 (The Beowulf Project 2006). Founded by Donald Becker and Thomas Sterling, the purpose of the

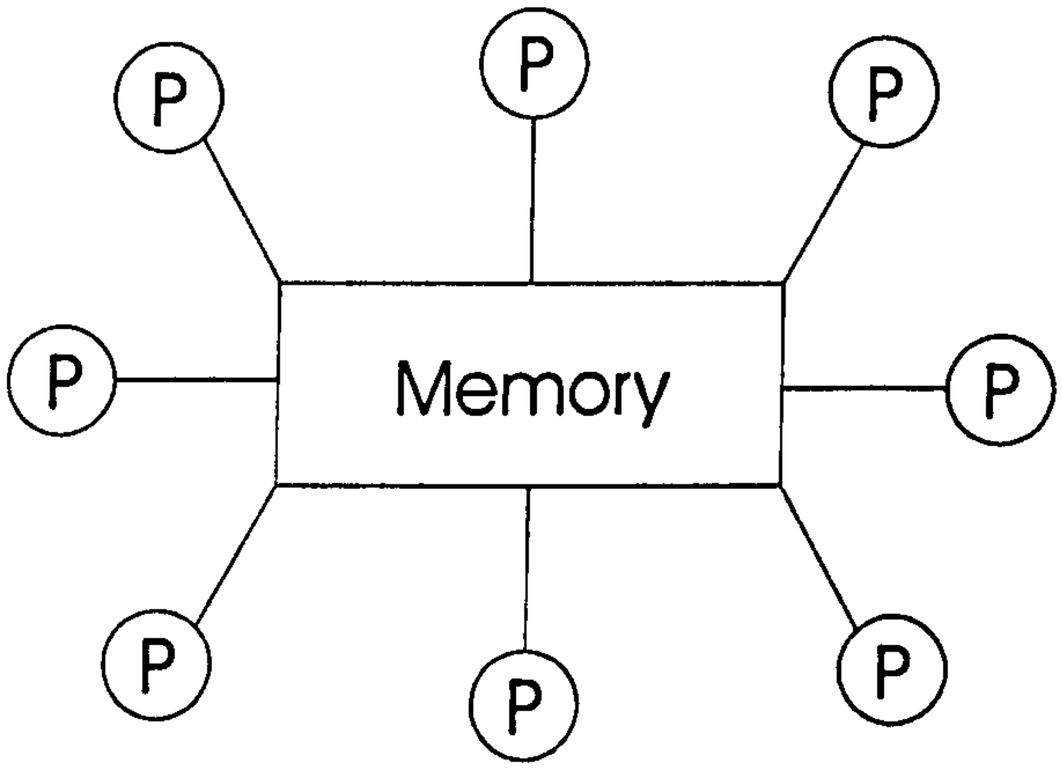


Figure 3.2: A Symmetric Multiple Processor system using a Shared Memory Architecture (Processors are denoted as P)

project was to provide High Performance Computing (HPC) clusters, built with commercial off-the-shelf hardware, for the specific purpose of performing tightly coupled HPC computations.

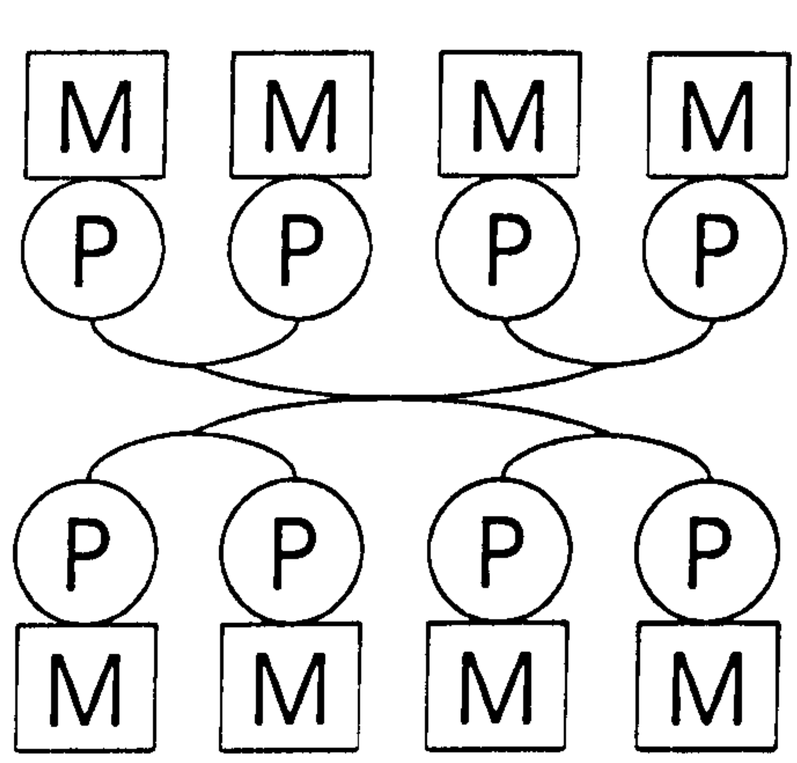


Figure 3.3: A Scalable Parallel Computing Architecture using Distributed Memory (Processors are denoted as P, Memory is denoted as M)

Distributed Computing

Around the same time as the development of distributed memory MPP systems Lenstra and Manasse (1989) were working on the concept of distributed comput-

ing. Lenstra and Manasse (1989) wrote software that distributed factoring tasks amongst computer workstations across the world by email. This software used idle CPU (Central Processing Unit) cycles to perform the computations. In 1994, Atkins *et al.* (1994) succeeded in factoring RSA-129, a 129 digit prime number issued as a challenge by RSA security, using the same technique. Since this work, the idea of using idle CPU cycles from a geographically distributed network of desktop computers has proved popular.

In 1997, the Distributed.net project (2006), a non-profit organisation dedicated to the advancement of distributed computing, was formed. Since its launch, distributed.net has successfully cracked RC5-56 and RC5-64 (encryption challenges posed by RSA security using 56-bit and 64-bit keys respectively). Other projects also using distributed computing include the SETI@home project (2006) and the ClimatePrediction.net project (2006) (both originally launched in 1999).

Metacomputing

Although key concepts in Grid computing (such as the idea of computing as a utility) can be traced back as far as the development of the Multics operating system (Corbató and Vyssotsky 1965) and the launch of ARPANET, the direct ancestor of the Grid is *Metacomputing* (Smarr and Catlett 1992). Metacomputing was a term used by Larry Smarr[3] to refer to:

> "a network of heterogeneous, computational resources linked by software in such a way that they can be used as easily as a personal computer." (Smarr and Catlett 1992)

Smarr and Catlett (1992) envisioned a national metacomputer linking supercomputer sites across the US in order to harness the processing power of multiple

[3]Director of the National Center for Supercomputing Applications (NCSA) from 1985-2000

supercomputers. In 1995 the Information Wide Area Year (I-WAY) project (DeFanti *et al.* 1996) attempted to realise this vision. The goal of the I-WAY project was to integrate existing high-bandwidth networks to enable researchers to use multiple supercomputers and advanced visualisation systems to conduct large-scale computations.

The I-WAY experiment linked seventeen supercomputer centres, five virtual reality research sites, and over 60 application groups using existing Asynchronous Transfer Mode (ATM) networks. A key part of the I-WAY system was the use of a dedicated point-of-presence (I-POP) machine at each of the sites, running a set of services to provide scheduling, security, parallel programming support, and a distributed file system (Foster *et al.* 1996). These I-POP machines greatly simplified the integration of sites into the I-WAY network, and provided uniform authentication, resource reservation, process creation, and communication function across all the resources in the network.

Metacomputing has many similarities with both parallel and distributed computing, but differs in important ways (Foster and Kesselman 1997). Like distributed computing, metacomputing seeks to integrate multiple resources of varying capabilities and in different administrative domains. However, typically metacomputing applications require high performance, thus requiring very different programming models. Metacomputing applications, like those in parallel computing, often need to schedule communication carefully to meet performance requirements. However, due to the heterogeneous and dynamic environments found in metacomputing systems, existing parallel computing techniques are often unsuitable.

Foster and Kesselman (1997) suggest that, while metacomputing can build on distributed and parallel computing technologies, significant advances are required to make metacomputing a reality. To this end they started the Globus project (The Globus Alliance 2006) which aims to provide a *metacomputing infrastructure toolkit*. The following sections provide an introduction to application development

in a Grid computing environment. Firstly the key functionality needed for the development of Grid computing applications is introduced, and then an overview of three representative applications of Grid computing is presented.

3.3 Services for Application Development on the Grid

The Grid computing paradigm is designed to make the compute resources that form the Grid available easily, uniformly, and transparently. The Grid can be represented as a layered architecture (see Figure 3.4) (Barnard *et al.* 1999), where the virtual machine layer provides a uniform interface to allow transparent access to the computational resources in the hardware layer. The Globus Toolkit (Foster and Kesselman 1999) facilitates the development of Grid-based applications by providing a set of core services that can be used to form this virtual machine layer. The Globus Toolkit takes a "bag-of-services" approach to creating *Grid-enabled* systems (Brunett, Czajkowski, Fitzgerald, Kesselman, Foster, Tuecke, Johnson and Leigh 1998), allowing applications to implement whatever services are required.

To provide uniform and transparent access to the resources in the hardware layer, the virtual machine layer must provide functionality for:

- Resource Monitoring and Discovery
- Secure Authentication
- Execution Management
- Data Management

The Globus Toolkit provides an open-source, community-based set of core services that help to implement the functionality required for this virtual machine

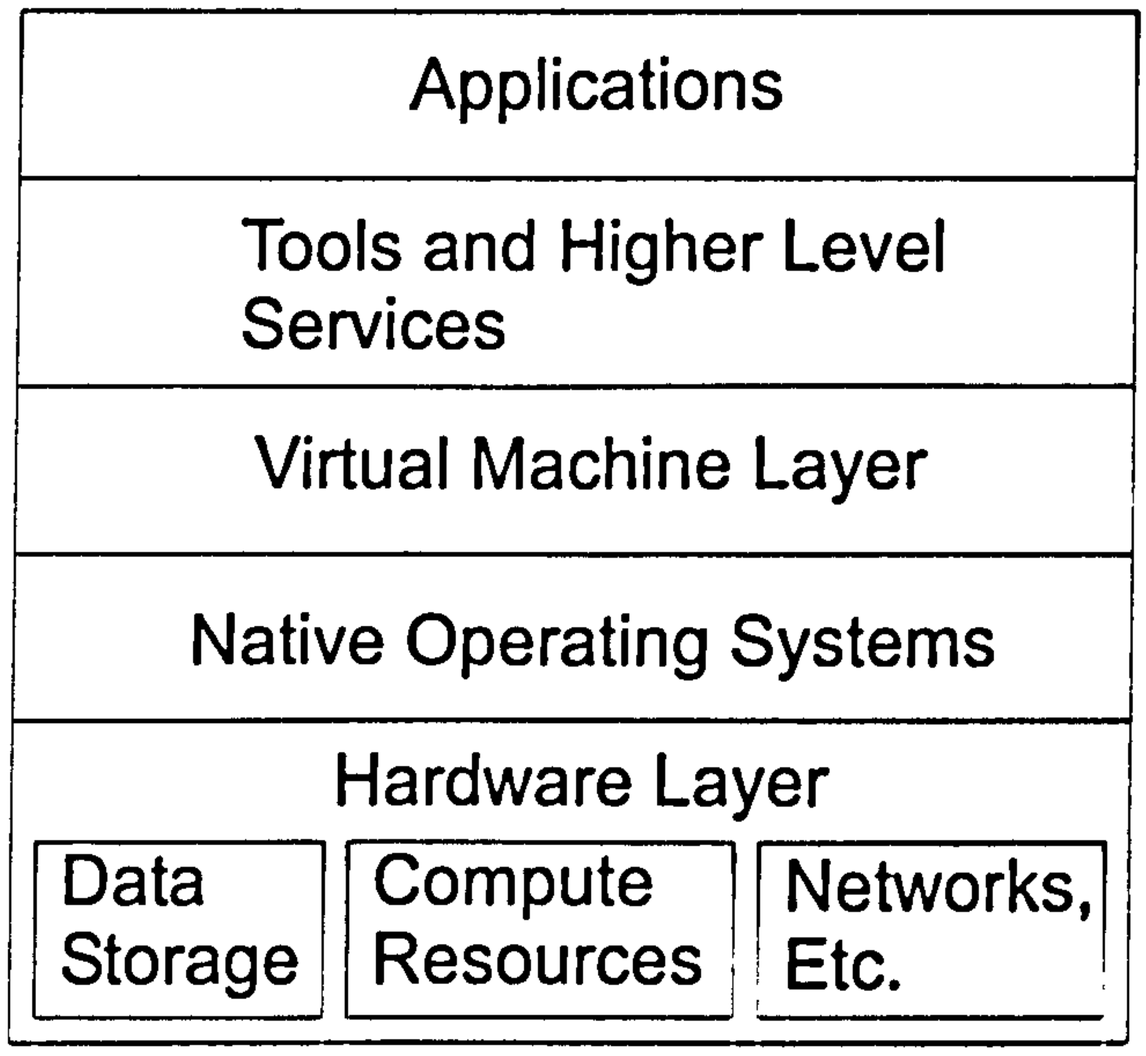


Figure 3.4: A Layered Grid Architecture

layer, and thus provide the basis for development of Grid-enabled applications. These core services help applications to discover appropriate resources, gain access to those resources in a secure way, deploy the application executable, and stage data in and out of the application. Since version 3, the Globus Toolkit has been based on the Open Grid Services Architecture (OGSA) introduced by the Globus Project. OGSA builds on current Web Service concepts and technologies (see below) to support the creation, maintenance, and application of ensembles of services (Foster *et al.* 2002).

Web Services

A Web Service is defined by the W3C as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages" (W3C Working Group 2004). Web Services are accessible through standards-based internet protocols such as HTTP and are enabled by

three core technologies (Chappell and Jewell 2002):

- Simple Object Access Protocol (SOAP)
- Web Services Description Language (WSDL)
- Universal Description, Discovery, and Integration (UDDI)

These technologies work together as shown in Figure 3.5 to provide functionality to an application. The Web Service client (i.e. the application) queries a UDDI registry to find a service providing the desired functionality. This can be done by service name, service category, or other identifier. Once this service has been located the client queries the WSDL document to find out how to interact with the service. The communication between client and service is then carried out by sending and receiving SOAP messages that conform to the XML schema found in the WSDL document.

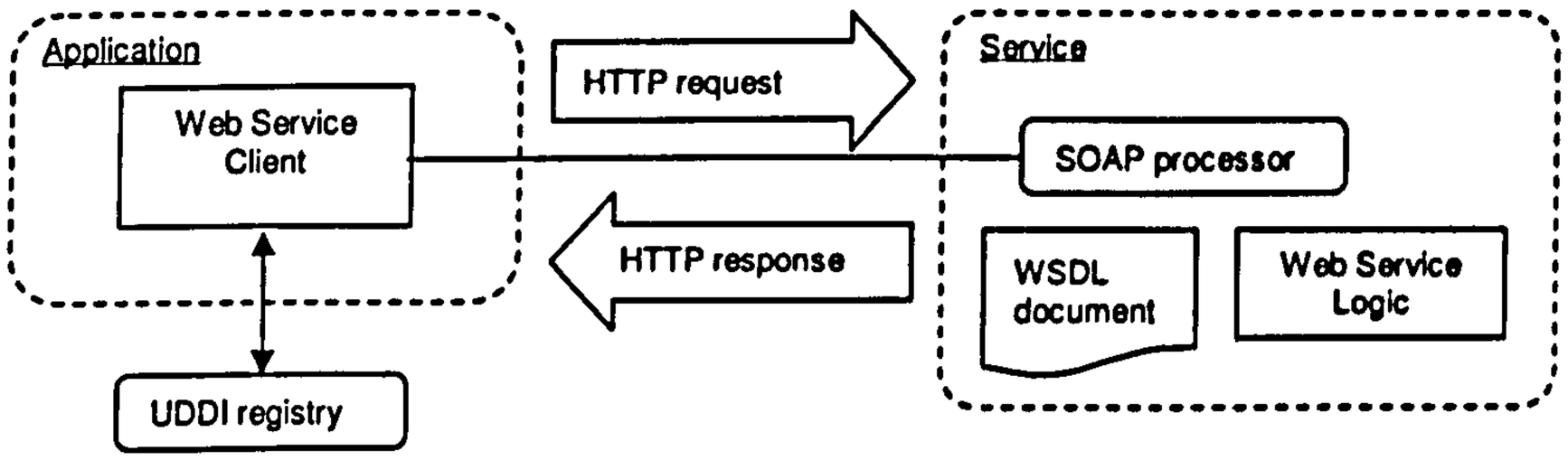


Figure 3.5: Interaction Between Core Web Service Technologies

Open Grid Services Architecture

The Open Grid Services Architecture represents computational resources, data resources, programs, networks and databases as services utilising the Web Service technologies mentioned previously. There are three main advantages to representing these resources as services:

1. **It aids interoperability.** A service-oriented view addresses the need for standard service definition mechanisms, local/remote transparency, adaptation to local OS services, and uniform semantics (Foster *et al.* 2002).

2. **It simplifies virtualisation.** Virtualisation allows for consistent resource access across multiple heterogeneous platforms by using a common interface to hide multiple implementations (Foster *et al.* 2002).

3. **It enables incremental implementation of Grid functionality.** The provision of Grid functionality via services means that the application developer is free to pick and choose the services that provide the desired behaviour to their application.

3.3.1 Resource Monitoring and Discovery

Resource monitoring and discovery are vital functions in a distributed computing environment (Foster 2005). They are particularly important when the resources in that environment are located at multiple geographically distributed sites. Resource monitoring is important in such situations to allow tasks such as fault detection and performance optimisation to be performed, whilst resource discovery allows application to identify resources or services with desired capability or functionality. However, Grid computing environments are dynamic, and the set of available resources may change frequently (Schopf *et al.* 2005). This dynamic nature of computational grids poses a significant challenge to the monitoring and discovery of resources.

A resource monitoring system for a Grid computing environment must be able to monitor the set of heterogeneous resources that comprise the hardware layer in Figure 3.4. It must also be scalable across geographically distributed sites, and have low latency because of the highly dynamic nature of the resources. The Global Grid Forum (GGF) performance working group have defined a standard architecture for Grid monitoring services that allows different tools to interoperate

(Tierney *et al.* 2002). It does this by defining three common interfaces and the way they interact with each other (see Figure 3.6). The producer and consumer components in Figure 3.6 generate and receive performance data respectively, whilst the directory service component is used in locating the available producers and consumers.

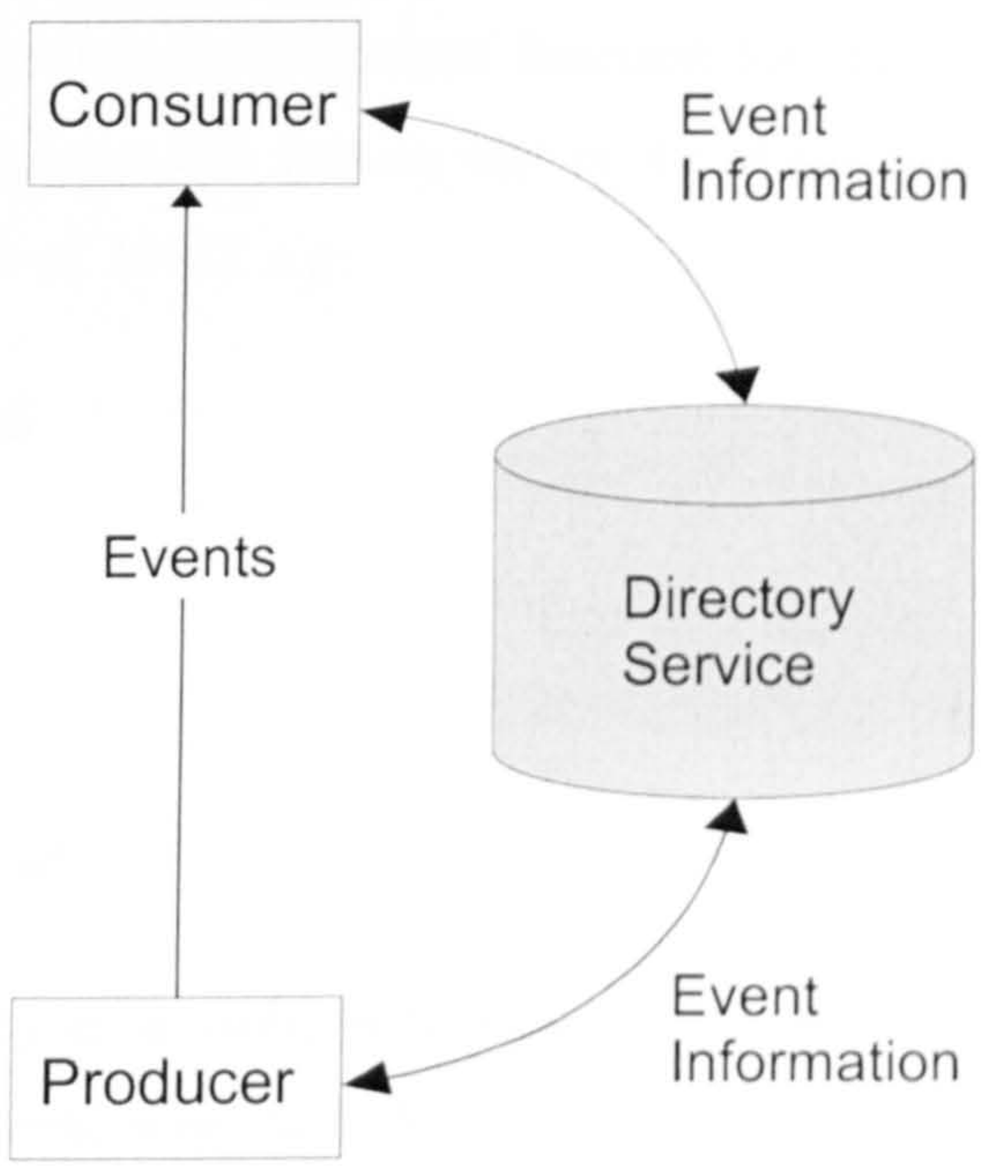


Figure 3.6: The Global Grid Forum Standard Architecture for Grid Monitoring Services

To dynamically discover resources in a Grid computing environment requires querying a registry of available resources. These registries must contain information such as the type of resource, how to interact with the resource, and the availability of the resource.

The Globus Toolkit provides resource monitoring and discovery by using the Monitoring and Discovery System (MDS). MDS provides a powerful framework for monitoring diverse collections of distributed resources and for obtaining information about these resources for the purpose of discovery (Foster 2005). It does this by providing aggregator services to collect information about the

state of resources (using both query and subscription models).

3.3.2 Secure Authentication

Grid computing environments are often characterised by geographically distributed resources spanning multiple administrative domains. This makes secure authentication an extremely important function for the virtual machine layer. Grid-based systems may also require any or all of the other standard security functions (Foster *et al.* 1998), e.g:

- Access Control
- Integrity
- Privacy
- Nonrepudiation[4]

A security policy in a multi-institutional *virtual organisation*[5] (VO) must interoperate with local security solutions whilst also providing the following (Foster *et al.* 2001):

- **Single Sign-on.** Users running complex applications may require access to many resources. Instead of having to re-authenticate for each resource, the user should be able to authenticate just once and then have access to multiple resources defined in the hardware layer (see Figure 3.4).
- **Delegation.** The user must be able to *delegate* their 'identity' to the application they are running. This application is then authorised to act on that users behalf, and to access any resources that the user is authorised

[4]Nonrepudiation means providing strong and substantial evidence that communications took place between two parties. This is usually provided by digitally signing communications.

[5]A Virtual Organisation refers to a dynamic set of individuals or institutions that agree to share Grid computing resources according to a set of sharing rules.

to access. The application should also be able to delegate a subset of this 'identity' to sub-tasks.

- **User-based Trust Relationships.** If a user is authorised to access resources located in multiple administrative domains, access to those resources must not require interaction between security policies at those domains.

Due to the complexity of the security problem, it is important that any solution is based on existing standards as much as possible (Tuecke 2001). The Globus Toolkit uses the Grid Security Infrastructure (GSI) to provide standards-based asymmetric encryption at both transport and message level (The Globus Security Team 2005). Transport level security involves encrypting the entire message, whilst message level security only encrypts the content and leaves the headers unaltered. GSI uses X.509 certificates to provide a standard format for secure identity management and distribution (Farrell and Housley 2002).

3.3.3 Execution Management

Once a Grid user has discovered appropriate resources for their application and securely authenticated themselves, they then need to be able to run tasks using those Grid resources. To do this they need to acquire access to the resources, configure them to meet the needs of the task, stage the executable for the task, initiate execution of the program, and monitor and manage the resulting computation (Foster 2005). The execution management component of the virtual machine layer must interoperate with local resource managers (for example, cluster management systems such as Sun Grid Engine and Platform LSF), and should also be able to interact with an information service to receive information about the current availability and capability of resources (see Figure 3.7).

The Globus Toolkit uses the Grid Resource Allocation and Management (GRAM) service to address the problem of resource and execution management.

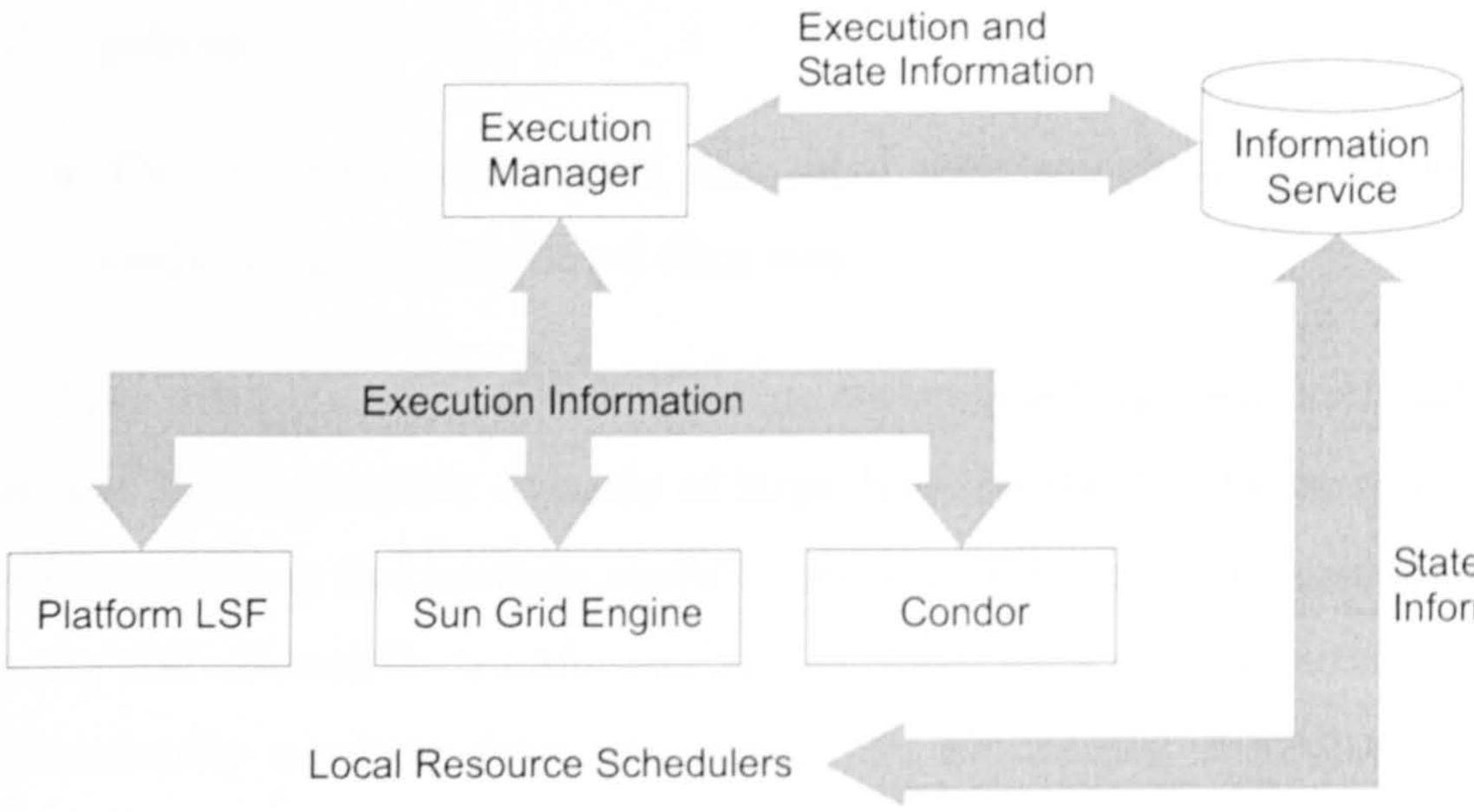


Figure 3.7: An Example of an Execution Management Service provided by the Virtual Machine Layer

GRAM provides a set of services for submission, monitoring and cancelling jobs on Grid computing resources. It is important to note that GRAM is not a resource scheduler; instead it provides a common interface to local resource schedulers (see Figure 3.7) (Czajkowski *et al.* 1998). GRAM uses the Monitoring and Discovery System (see Section 3.3.1) to receive information about the state of resources in the Grid.

3.3.4 Data Management

Grid based applications, such as complex computer simulations and data analysis, often need to access and share large data sets located across multiple sites. Data management in such environments presents significant challenges, due mainly to the following (Atkinson *et al.* 2004):

- The diverse data usage scenarios encountered in a Grid computing environment (i.e. dealing with both binary formatted and relational data).
- The heterogeneity of data resources encountered in a Grid computing environment (i.e. diverse storage systems, data formats, and data access

policies).

- The performance demands associated with accessing, manipulating and analysing large distributed data sets.

Data management in Grid computing environments involves both *data transfer* and *data replication.* Transfer of large data sets (both between multiple data storage resources and between applications and data storage resources) requires a secure and efficient file transfer mechanism. Security is needed due to potentially commercially sensitive data sets, whilst the file transfer mechanism must be efficient, as some data-intensive scientific applications may require terabytes of data to be transferred (Allcock *et al.* 2002). Data replication capabilities are needed to allow copies to be made on local resources (see Figure 3.8 for a typical high-level data replication scenario), thus reducing access latency and allowing local access controls to be exerted over this data (Atkinson *et al.* 2004).

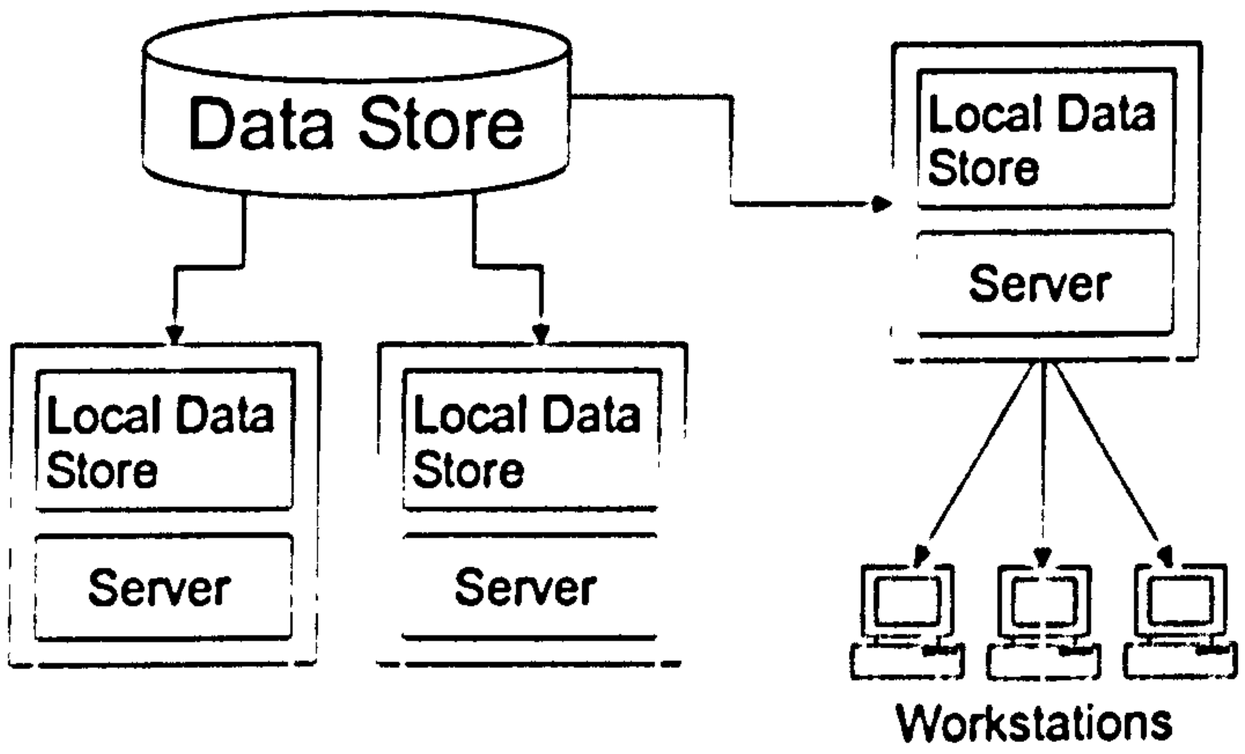


Figure 3.8: A Typical Data Replication Scenario

Data management in the Globus Toolkit is primarily provided by:

- **GridFTP.** GridFTP provides a high-performance, secure, and reliable data transport mechanism.

- **Reliable File Transfer (RFT) Service.** The RFT service provides a web-service for managing multiple GridFTP transfers reliably.

- **Replica Location Service (RLS).** RLS provides a scalable registry for maintaining and locating replicated data sets.

The Globus Toolkit also enables access to structured and relational data via the Open Grid Services Architecture - Data Access and Integration (OGSA-DAI) tools developed within the UK e-Science program (Karasavvas *et al.* 2004).

3.4 Applications of Grid Computing

3.4.1 The Synthetic Forces Express Project

Synthetic Forces Express (SF-Express) (Messina *et al.* 1997) is a highly scalable implementation of the Modular Semi-Automated Forces (ModSAF) Distributed Interactive Simulation (DIS). Due to their ability to model the movement and behaviour of large numbers of complex entities, distributed interactive simulations are frequently used by the military for training, planning and analysis purposes (Brunett, Davis, Gottschalk, Messina and Kesselman 1998). These entities are typically high-fidelity, distributed, and support Human-In-Loop (HIL) interactions.

The number of entities used in previous DIS applications was limited by both the available computational power and network bandwidth (Messina 1999). Typically these DIS applications have been run on distributed workstations connected by local area networks, with communications between work stations performed through the exchange of Protocol Data Units (PDUs) (Brunett, Davis, Gottschalk, Messina and Kesselman 1998). As the number of simulated entities grows, the amount of processing power and network bandwidth needed can easily exceed what is available.

SF-Express has been adapted to execute in a Grid computing environment by incrementally incorporating components from the Globus Toolkit (Brunett, Czajkowski, Fitzgerald, Kesselman, Foster, Tuecke, Johnson and Leigh 1998). This approach meant that the application did not have to be re-written to operate on the Grid; instead Globus Toolkit components were integrated to address specific problems or provide desired functionality. For example, the MDS and GRAM components (see Section 3.3.1 and Section 3.3.3 respectively) greatly simplify the start-up of the SF-Express application. MDS provides an automated approach to system configuration whilst GRAM provides a simple, uniform interface to the local resources schedulers. This is a significant improvement over the previous approach of hand-written scripts based on detailed static resource information (Brunett, Davis, Gottschalk, Messina and Kesselman 1998).

3.4.2 The National Fusion Collaboratory Project

The National Fusion Collaboratory (NFC) project aims to advance scientific understanding in magnetic fusion research by enabling more efficient use of community resources (Keahey *et al.* 2002). Magnetic Fusion research is concerned with sustaining a nuclear fusion reaction in a plasma contained by very strong magnetic fields. The long-term goal is to use this technology to develop fusion reactors. Experimental data produced by the fusion community is an important resource to researchers, with over 250MB of data produced for each pulse[6] (Keahey *et al.* 2002). Another important set of resources in magnetic fusion research are the realistic non-linear 3D plasma models produced by the theoretical and simulation community (Schissel *et al.* 2004). The National Fusion Collaboratory aims to provide transparent access to remote computational resources, experimental data, and simulations to allow more effective comparisons

[6]Magnetic fusion experiments operate in a pulsed mode, producing pulses of plasma of up to 10 seconds duration. These pulses occur every 10 to 20 minutes, with 25 to 35 pulses per day (The National Fusion Collaboratory Project 2001).

between theory and experiment (Keahey *et al.* 2002).

The National Fusion Collaboratory project provides a good example of a *virtual organisation* (see Footnote 5). The project aims to provide collaborative software tools for use by the fusion research community, as well as sharing Grid resources located at multiple, geographically distributed, sites spanning different administrative domains (Burress *et al.* 2005). This collaboration is enabled by using components from the Globus Toolkit, and other distributed computing technologies, to provide essential services for the operation of a virtual organisation.

The NFC provides single sign-on capabilities using the Grid Security Infrastructure (see Section 3.3.2), providing secure transparent authorisation across multiple administrative domains. This single sign-on capability is important in a virtual organisation (such as the NFC) because it allows members of the VO to run applications that require access to remote data sets and remote compute resources without having to re-authenticate for each remote resource used (Burress *et al.* 2005). Another Globus Toolkit component used by the National Fusion Collaboratory is GRAM (see Section 3.3.3). GRAM is used to interact with local schedulers on the remote resources in the virtual organisation, to provide a simple protocol for secure remote submission, monitoring and management of compute jobs. The NFC project has also developed a Resource-Oriented Authorization Manager (ROAM) that allows resource owners to formulate community-specific rules on access. GRAM can then be configured to use these rules (Burress *et al.* 2005).

Data Access in the NFC project is provided using a GSI enabled version of MDSplus[7] combined with an SQL database. This provides secure access to data sets, with the ability to efficiently search using calls to the relational database.

[7]MDSplus is a set of software tools for data management developed jointly by the Massachusetts Institute of Technology, the Center for Nuclear Resarch in Italy, and the Los Alamos National Laboratory. MDSplus is the most widely used system in the magnetic fusion energy program (MDSplus project 2006).

Output from analysis and simulations can also be retrieved via GridFTP (see Section 3.3.4) if an MDSplus server is not available (Schissel 2005).

Interaction and collaboration between researchers and research groups in the National Fusion Collaboratory is also enabled by Access Grid technologies. The Access Grid provides a multi-site collaborative experience integrated with high-end visualisation environments to enable researchers to share results and perform complex data analysis between remote locations (Schissel *et al.* 2004).

3.4.3 The Distributed Aircraft Maintenance Environment Project

Aero-engines (Rolls-Royce 1986) are extremely reliable machines, and operational failures are rare. However, due to a shift in emphasis within aero-engine companies to power-by-the-hour contracts[8], great effort is being put in to reducing delays and cost of ownership of aircraft using advanced engine health monitoring techniques (Shenfield *et al.* 2005). To support these fault diagnosis and prognosis techniques, much more detailed operational data must be collected from the engines. As a result of this, each engine on a civil airliner may produce up to 1GB of data per flight. Rolls-Royce currently has over 50,000 engines in service, with total operations of around 1M flying hours per month. It is therefore easy to envision many 100s of GBs of data being transmitted and analysed every day (Ong *et al.* 2005).

The Distributed Aircraft Maintenance Environment (DAME) is a £3M UK e-Science pilot project involving Rolls-Royce, Data Systems and Solutions, and other commercial and academic partners, investigating the use of Grid computing technologies to provide a *virtual workbench* for advanced fault diagnosis and prognosis of aircraft engines (Austin *et al.* 2004). Grid computing environments

[8]Power-by-the-hour contracts involve airlines making regular fixed payments based on the hours flown by an engine and, in return, the manufacturers of the engine retain the responsibility for servicing and maintenance.

provide an infrastructure capable of managing the large amounts of operational data produced by these engines and performing compute intensive modelling and analysis to identify current and potential faults.

The DAME system provides core services for fault diagnosis and prognosis, including a pattern matching service for data mining of large-scale data sets, an engine modelling service to provide model-based fault diagnosis and isolation capabilities, and a case-based reasoning service to provide knowledge based maintenance advice. These software tools, along with the operational engine data and domain expertise, must be geographically distributed to cope with the demands of an international operation (see Figure 3.9, reproduced from Shenfield *et al.* (2005)). It is also important that, due to the commercially sensitive information involved, access to services and data should be restricted to authorised members within the organisation. The Grid Security Infrastructure (see Section 3.3.2) provides secure authentication to these services using X.509 certificates, and also enables encrypted communication at both transport and message level.

Data access and data management in the Distributed Aircraft Maintenance Environment is provided by the Storage Resource Broker (SRB) (Baru *et al.* 1998) developed at the San Diego Supercomputer Center. SRB allows the distributed data storage resources in the DAME system to be treated as a single logical file system, and supports the management, controlled sharing, replication and transfer of those distributed data sets. Execution management for the services in DAME is provided by custom developed solutions interacting with the Sun Grid Engine scheduler running on the White Rose Grid compute resources (see Section 4.3).

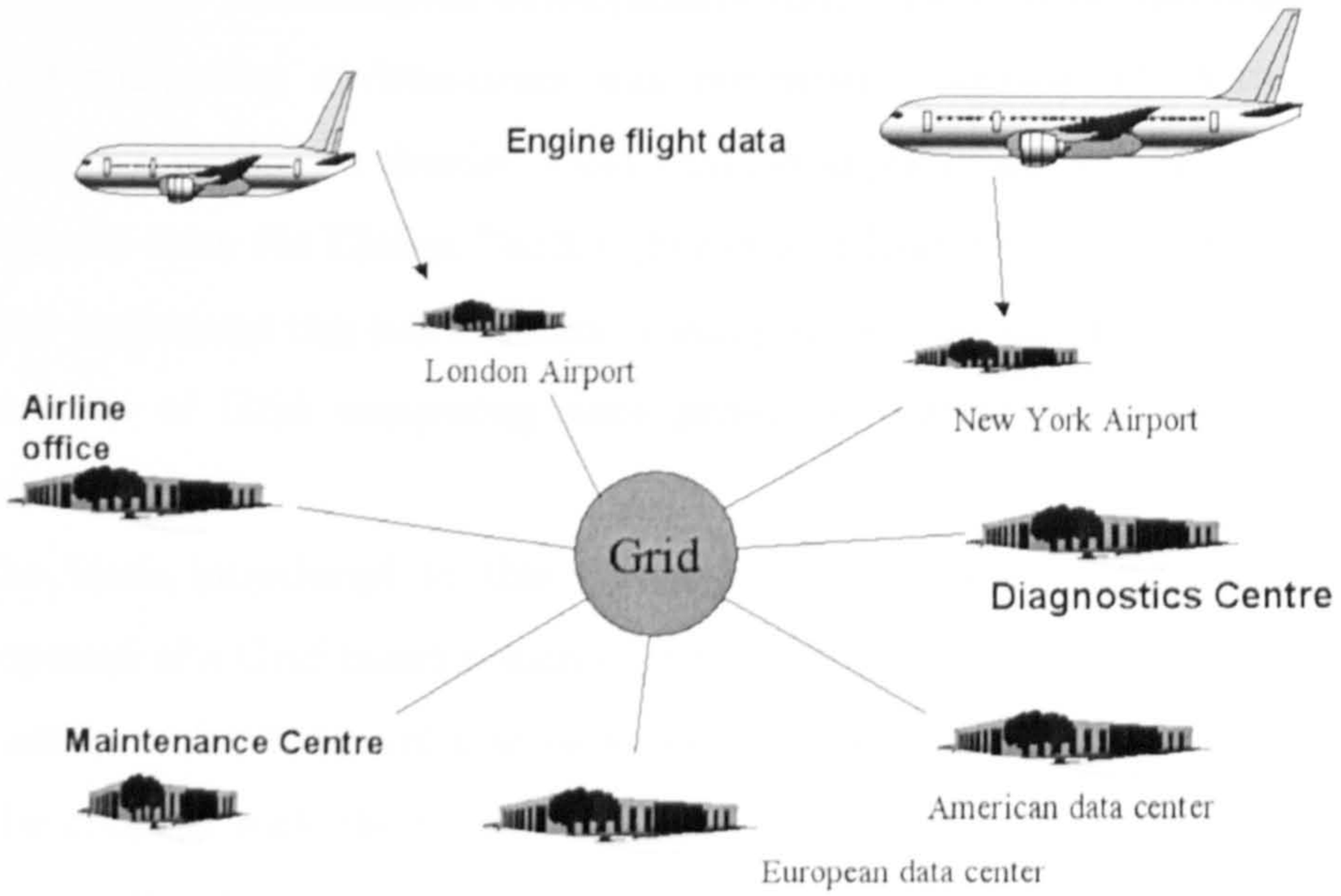


Figure 3.9: The Distributed Aircraft Maintenance Environment Architecture

3.5 Summary

As mentioned earlier, modern science and engineering is increasingly concerned with performing complex computer simulations and data analysis. Often these tasks require larger amounts of compute resources than are available in a single place. However, advances in networking and computer technologies have enabled the creation of *computational grids* where researchers across multiple geographically distributed sites can share compute resources (including data and data storage resources) transparently. This emerging paradigm of Grid computing therefore offers many benefits in both science and engineering. In the engineering design domain especially, drastically reducing the time taken to obtain useful results from the optimisation of complex real-world problems is of great practical benefit - potentially reducing the time to market for many products.

This Chapter has aimed to provide a thorough review of the Grid computing paradigm from an application development perspective. In Section 3.2 a historical

overview of the technological developments that have enabled the construction of Grid computing environments was presented. Section 3.3 described the core services needed to develop Grid enabled applications and introduced key components from the Globus Toolkit (Foster and Kesselman 1999) that help the designer implement this functionality. Finally, in Section 3.4, three representative applications of Grid computing were presented and their key features were outlined.

The ideas introduced in this Chapter will be used in Chapter 4 in the development of a Grid-based resource broker that will allow an application to run tasks across multiple administrative domains. This multi-site resource broker will then be coupled with the Service Oriented Architecture approach to providing Grid computing functionality outlined in this Chapter to provide job submission services for the Grid-enabled evolutionary algorithm introduced in Chapter 5.

Chapter 4

Development of a Resource Broker for Grid-Enabled Applications

4.1 Introduction

Resource allocation and scheduling in computer systems is performed at many levels. For instance, the process of opening a web page on a personal computer requires the computer's operating system (OS) to allocate processor time and memory to the web browser application so it can run. Resource allocation and scheduling becomes more complicated when the scheduler has to be concerned with multiple users attempting to run applications using multiple processors, such as in a massively parallel processor system or compute cluster (see Section 3.2.2). Several Resource Management Systems (RMSs) exist to address the problem of scheduling at this level, and a survey of some of these systems is presented in Section 4.2.

As discussed in Chapter 3 earlier, engineers and scientists are increasingly concerned with complex simulation and data analysis tasks that frequently require

larger amounts of compute resources than are available at a single site. As a result of this, there is a growing trend towards scalable and distributed computing technologies (see Section 3.2.2), and in particular the emerging paradigm of Grid Computing.

Computational Grids often consist of many disparate heterogeneous resources located across multiple administrative domains. Making resource allocation and scheduling decisions in such environments is a difficult task. One key challenge is that the resources in a Grid computing environment are usually owned by many different stakeholders, thus a Grid-based scheduler cannot take control of the available resources (Schopf 2002). Another challenge is the dynamic nature of computational Grids. The availability and capacity of the resources in a computational Grid constantly changes, and the resource scheduler must therefore make a "best guess" at where to allocate jobs based on potentially incomplete information.

Berman (1999) classifies Grid-based schedulers according to their focus:

- Resource-level scheduling aims to maximise the utilisation of resources in the system (usually in an equitable manner).
- Job-level scheduling aims to maximise the throughput of jobs in the system.
- Application-level scheduling aims to maximise the performance of an individual application by optimising application centric measures such as execution time.

Berman (1999) also notes that the goals of each of these categories of resource scheduler may be in conflict with each other. For example, resource-level and job-level schedulers focus primarily on the performance of the system, whilst application-level schedulers focus on application performance.

This Chapter provides an application-centric solution to the problem of resource management in computational Grids. Some existing solutions for

resource management are examined in Section 4.2 and, in Section 4.3, a production quality Grid computing environment (the White Rose Grid) is presented. Section 4.4 introduces a novel design pattern based framework for resource management developed as part of this thesis. A key component of any resource management system is its scheduling policy, and Sections 4.5 and 4.6 propose a novel workload allocation algorithm, coupled with an innovative hybrid approach to job scheduling, to enable the efficient distribution of tasks in dynamic, heterogeneous computational environments.

4.2 Survey of Resource Management Systems

Resource management systems in Grid computing environments can be classified by what level of control the RMS has over the resources it manages. Many resource management systems have *centralised control* over the resources that they manage. However, according to Foster (2002a), computer networks managed by these RMSs cannot be classified as true computational Grids (although they can constitute powerful Grid resources).

Grid-based meta-scheduling architectures have been proposed to overcome the need for centralised control over Grid resources (Schopf 2002, Vadhiyar and Dongarra 2002). These meta-schedulers must make scheduling decisions involving multiple geographically distributed heterogeneous resources. Grid-based meta-schedulers need to discover and evaluate available resources that are often located in different administrative domains, allocate jobs to appropriate resources, and manage execution of those jobs. To accomplish this, meta-schedulers typically interact with local resource management systems. Most local RMSs provide access to status information and job submission and monitoring services.

This Section will present a simple taxonomy of some of the most widely used resource management systems commonly found in Grid computing environments based on the level of control the RMS has over the available resources, and provide

an overview of the RMSs mentioned. The interested reader is also directed to Krauter *et al.* (2002) for further information.

4.2.1 Resource Management Systems with Centralised Control

Sun Grid Engine

Sun Grid Engine (SGE) (The Sun Grid Engine Project 2007) provides a powerful resource management system for compute clusters and campus Grids. Two versions of SGE are available: a commercial version (known as N1 Grid Engine) and an open-source version. The Grid Engine software is a centralised RMS and uses *enterprise resource policies* to manage workload allocation amongst the available compute resources (Sun Microsystems Inc. 2006). SGE provides high-level usage policies that can be customised by the site administrator according to the goals of that organisation. For example, usage policies can be customised to maximise throughput or resource utilisation.

SGE allows users to submit batch, parallel or interactive jobs. These are submitted to the *Master Host* which controls the job queues and scheduling activities. Multiple queues can exist in the system and jobs are allocated to these queues based on usage policies and job requirements. Once a job gets to the head of its queue it is assigned to an *Execution Host* which has permission to run the job (Sun Microsystems Inc. 2006).

Foster (2002b) does not classify a computer network managed by Sun Grid Engine as a true computational Grid because of the centralised control SGE has over the resources it manages. Sun Grid Engine is not capable of managing compute resources located in multiple administrative domains or across organisation boundaries; however, compute clusters and campus Grids managed by SGE can constitute powerful Grid resources (Foster 2002b).

Portable Batch System

The Portable Batch System (PBS) was originally developed for NASA because the scheduling capability of other batch systems at that time was inadequate for their growing interest in parallel computing (Henderson 1995). PBS was designed not only to support a range of system types (including both dedicated supercomputers and clusters of workstations), but also to allow the implementation of site specific scheduling policies by separating the scheduling functionality from the rest of the system. This separation allows sites the flexibility to implement custom scheduling policies that address site-specific needs.

PBS provides similar functionality for job submission and management as other batch systems, with clients making batch job submissions to a central job server which manages the queues in the system and sends the jobs for execution. Like SGE (see above), PBS also exerts centralised control over resources and so Foster (2002b) does not classify PBS managed computer networks as true computational Grids.

Load Sharing Facility

Platform's Load Sharing Facility (LSF) provides a set of commercial components to manage networks of heterogeneous workstations (Platform Computing Corporation 2007). The LSF Base component provides basic services for querying resources, selecting hosts, and job submission and execution. On top of this layer is the LSF Batch component which extends the functionality provided by LSF Base to provide batch scheduling services, as well as load balancing and policy-driven resource allocation (Platform Computing Corporation 2002).

LSF provides a powerful RMS for managing resources in a local cluster. However, like SGE and PBS (see above), LSF manages these resources centrally and cannot manage resources across administrative boundaries. For this reason resources managed by LSF are not classified as true computational Grids

(Foster 2002b). To address this need to cross administrative boundaries, Platform have developed LSF MultiCluster. LSF MultiCluster allows access to multiple, geographically distributed, clusters whilst also preserving local ownership and control of these resources. Foster (2002b) classifies LSF MultiCluster as a true Grid RMS.

4.2.2 Meta-Scheduling Architectures for Grid Computing

Globus Toolkit

The Globus Toolkit (Foster and Kesselman 1999) provides services to assist with resource monitoring and discovery, security, execution management, and data management (see Section 3.3) in Grid computing environments. The Globus Toolkit is not a resource management system *per se*; instead, using the the Grid Resource Allocation and Management service, it provides a common interface to local resource management systems (Czajkowski *et al.* 1998). Components from the Globus Toolkit are used in Condor-G to allow interaction between Condor and local RMSs (Frey *et al.* 2001).

Condor

The Condor project (The Condor Project 2007) aims to support High Throughput Computing (HTC) by providing a resource management system capable of both managing dedicated compute clusters and *cycle scavenging* in a distributed computing environment. Cycle scavenging allows organisations to harness CPU cycles from idle workstations and other compute resources in accordance with usage policies specified by the resource owner. For example, Condor can be configured to only allow job submissions to a workstation if there has been no keyboard or CPU activity for 15 minutes (Thain *et al.* 2003). Livny and Raman (1999) have recognised that for resource owners to donate their machines for use in an HTC environment requires that their rights are protected. This means

that these resources can only be used *opportunistically* and should be able to be reclaimed by their owner at any time.

Condor performs matchmaking between jobs and resources using *ClassAds* (Raman *et al.* 2000) to describe resource requests and resource offers. Resource offers are made by available machines in a Condor pool advertising their capabilities (i.e. available RAM, OS, CPU architecture) to a matchmaker. When a user submits a job to the system a resource request detailing the job requirements is passed to the matchmaker, and, if a resource is available that meets those requirements, the job is run on that machine.

Condor can be classified as a Grid resource management system as it can support resource sharing across organisational boundaries. This can be performed between Condor pools located in different administrative domains using *flocking* (Epema *et al.* 1996), or between a Condor pool and another resource management system using Condor-G (Frey *et al.* 2001). Condor-G uses the Globus Toolkit (see above) to communicate with local resource management systems, and is used in the European DataGrid project (The European DataGrid Project 2007) to perform job submission (Klous *et al.* 2006).

AppLeS

The Application Level Scheduler (AppLeS) project (Berman *et al.* 1996) provides an *application centric* approach to resource allocation and scheduling in distributed heterogeneous environments. In this approach, each Grid-based application has its own scheduler which determines a customised schedule for that application based on how it is affected by the state of various resources in the system (Berman 1999). The goal of each AppLeS agent (i.e. application specific scheduler) is to maximise the performance of its application.

AppLeS agents aim to schedule applications using similar principles as end-users do (Berman *et al.* 1996). However, the AppLeS agents can utilise real-time information in making scheduling decisions. Dynamic information

can be provided to an AppLeS agent using the Network Weather Service (Wolski 1996). AppLeS is not a true resource management system, it is an *application management system* which interacts with local RMSs such as SGE, PBS, or LSF (see Section 4.2.1) to provide multi-site application scheduling.

Silver

Silver is a reservation based meta-scheduler that allows distributed workloads to be run across multiple independent clusters (Cluster Resources Inc. 2004). It interacts with the local resource management system running on each resource, and allows full autonomy for the resources it utilises (i.e. it respects all local policies and priorities). Silver is currently used to manage the resources in the Cluster Ohio Grid (Ohio Supercomputer Center (OSC) 2007) at the Ohio Supercomputer Centre. However, the Silver meta-scheduler currently only provides limited support for many commonly used local resource management systems such as SGE and LSF.

GridWay

The GridWay meta-scheduler uses the Globus Toolkit (see above) to share computing resources managed by a variety of local RMSs (Distributed Systems Architecture Group, Universidad Complutense de Madrid 2007). The use of the Globus Toolkit to mediate communication between local resource management systems and the GridWay meta-scheduler means that GridWay provides a lightweight meta-scheduling solution. However, it also means that the GridWay meta-scheduler requires the Globus Toolkit to be installed on all the resources available to it. It also cannot provide resource management features that are not available in GRAM (such as advanced reservation of resources).

4.3 The White Rose Grid

The White Rose Grid (The White Rose University Consortium 2007) is a multi-institutional computational Grid launched in 2002 by the universities of Sheffield, York and Leeds. The main objective of the White Rose Grid project is to support e-Research by providing users with access to large amounts of heterogeneous compute resources. The White Rose Grid currently consists of five high-performance compute nodes located at three different sites (see Figure 4.1), and in 2003 was awarded the status of e-Science Centre of Excellence.

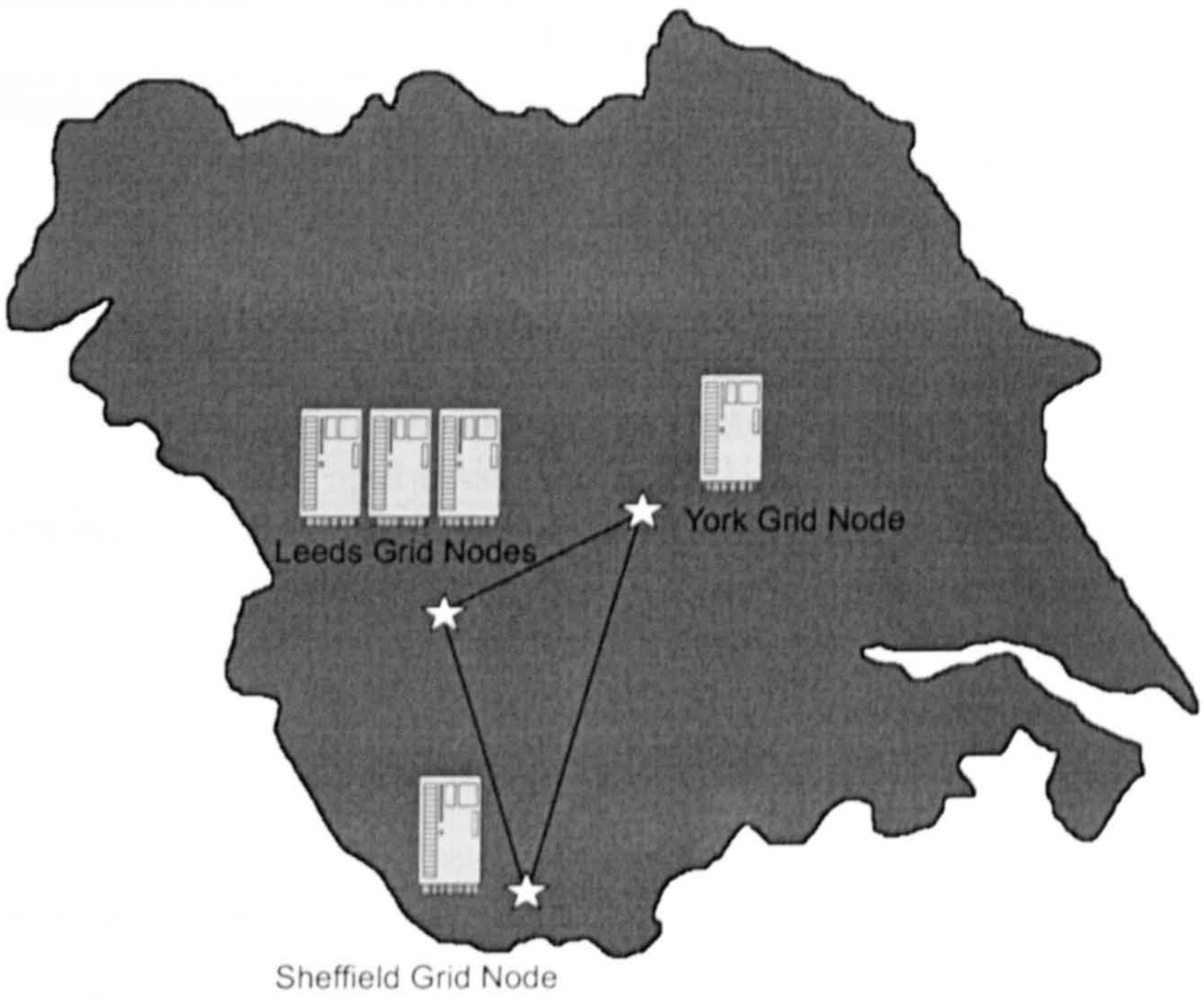


Figure 4.1: The Network Topology of the White Rose Grid

The three participating institutions in the White Rose Grid consortium reserve 75% of their Grid resources for users within their institution and allocate the remaining 25% to other users of the Grid. The current Grid resources available are outlined in Table 4.1. These resources are connected by the high bandwidth Yorkshire and Humberside Metropolitan Area Network (YHMAN),

Sheffield	
Iceberg	Iceberg is a compute cluster consisting of 320 2.4GHz AMD Opteron processors running the Scientific Linux operating system. 160 of these processors are available for general use, whilst the other 160 are reserved for the GridPP project (The GridPP Project 2007).
York	
Pascali	Pascali consists of a large memory cluster for jobs with big memory requirements and a Beowulf cluster (see Section 3.2.2) with 24 nodes. The Beowulf cluster is based on dual core AMD Opteron processors, with each node comprising of two processors. These clusters run the Scientific Linux operating system.
Leeds	
Maxima	Maxima is a constellation of shared memory SMP systems (see Section 3.2.2) based on Sun Fire 6800 and V880 servers. This node offers 60 UltraSPARC III processors running the Solaris operating system.
Snowdon	Snowdon is a cluster of 128 dual Intel Xeon processor based compute nodes. This Grid node offers a total of 256 processors for dedicated batch use and runs the Linux operating system.
Everest	Everest is a cluster based on AMD Opteron dual core processors running the Linux operating system. It consists of 66 dual processor Sun V20z servers and 8 quad processor Sun V40z servers.

Table 4.1: White Rose Grid Resources

and are managed at a local level by Sun Grid Engine (see Section 4.2.1). However, there is no meta-scheduler currently available to allow a scientist or engineer to transparently utilise all these multi-institutional resources.

4.4 Design Patterns for Meta-Schedulers

Design Patterns (Gamma *et al.* 1995) are widely used in software engineering to provide reusable templates for solving commonly occurring problems in software

design. The concept of design patterns originated in the architectural system proposed by Alexander *et al.* (1977) which consisted of a pattern language describing proven solutions to common architectural design problems. The idea behind design patterns is that:

> "each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice." (Alexander *et al.* 1977)

Software design patterns can speed up the development process by making it easier to reuse successful designs and architectures (Gamma *et al.* 1995). The use of patterns can also help improve the quality of code and reduce programming errors by making the designer explicitly aware of potential issues and trade-offs inherent in their design choices.

As the field of Grid computing matures, and more experience is gained in the development of Grid-enabled applications, design patterns are attracting a growing interest from the Grid community (Katsiri *et al.* 2006). Application designers are now finding common problems occurring in Grid-enabled software development, such as managing jobs and locating resources, and are developing design patterns to address these problems. In the following Section several patterns addressing the problem of resource management in Grid-enabled applications will be introduced. These patterns follow the format introduced in Gamma *et al.* (1995), presenting the following attributes:

- Pattern Name.
- Intent.
- Motivation.

- Applicability.
- Structure.
- Participants and Collaborations.
- Consequences.
- Related Patterns.

Unified Modelling Language (UML) diagrams (Rumbaugh *et al.* 1999) are used to illustrate certain attributes in the following design patterns. The structure of each system is shown by a UML class diagram. These emphasize what components need to be present in the system, as well as showing the associative relationships between components[1]. UML sequence diagrams are also used where appropriate to illustrate interactions between the components in a system by showing the exchange of messages between them.

[1]Associations between UML entities (i.e. components in the system) are shown by solid lines (links) between them. These links also often have a multiplicity value showing the number of entities that can be associated with a single entity on the other end of the link. In this notation, * represents the possibility of many entities being associated with a single entity from the other side.

Resource Broker Pattern

Intent:

The Resource Broker pattern provides an application with the ability to discover and use Grid resources. It will aggregate other services to do this.

Motivation:

A computational Grid typically consists of a dynamically changing set of geographically distributed compute resources. These resources are often situated across multiple administrative domains, with each resource being controlled by a local resource management system. A Grid-enabled application should have the ability to discover and use appropriate compute resources. The solution is to provide a resource broker agent that can discover these resources on behalf of the application and manage the execution of application subtasks across these resources.

Applicability:

The Resource Broker pattern should be used when:

- the application wishes to locate appropriate resources for execution.
- the resource pool that the application is authorised to use is dynamic in nature.
- the application wishes to execute subtasks across multiple resources in the Grid.

Structure:

The structure of the Resource Broker is shown in Figure 4.2.

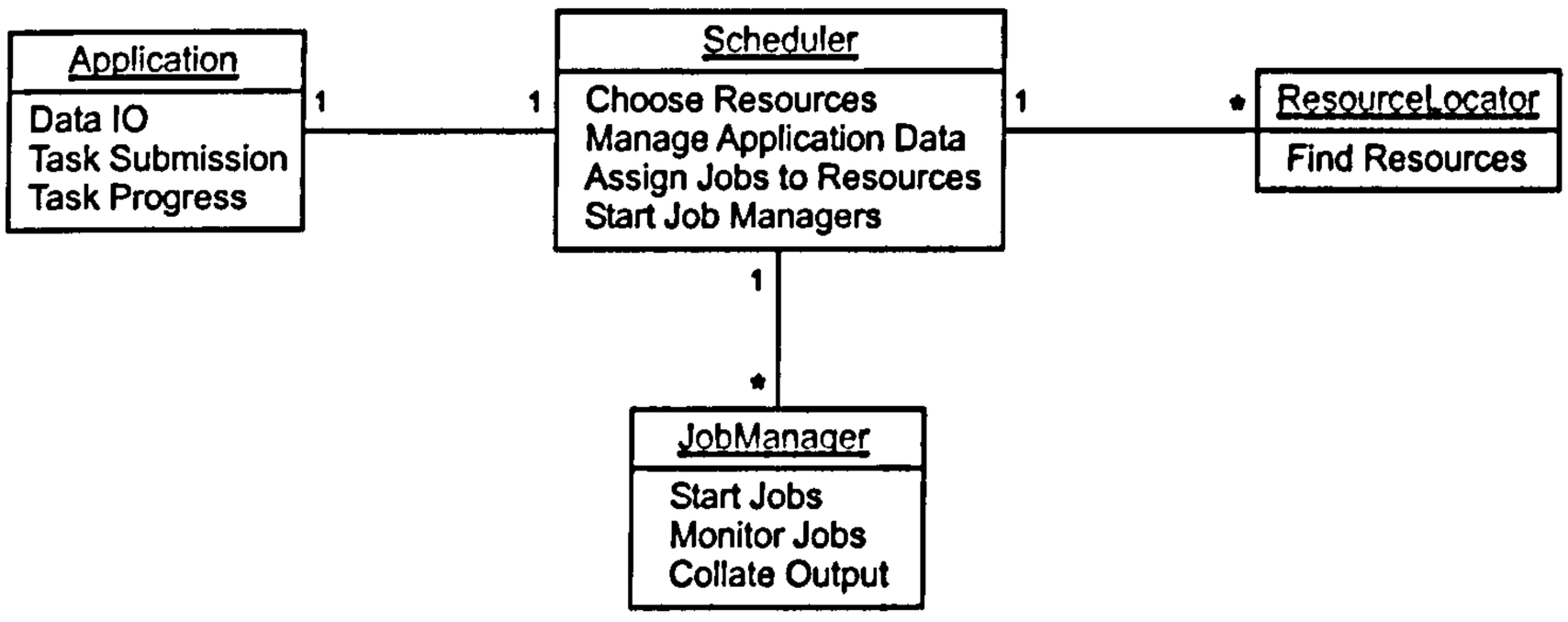


Figure 4.2: A UML Class Diagram for the Resource Broker Pattern

Participants:

- **Application**
  - passes data and subtasks to the Scheduler.
- **Scheduler**
  - discovers resources using the ResourceLocator.
  - assigns subtasks to remote resources.
  - manages Application data.
  - starts JobManagers on remote resources.
- **ResourceLocator**
  - finds appropriate resources for use by the Application.
- **JobManager**
  - runs the jobs on the remote resources.
  - manages the staging of data for jobs.
  - monitors jobs.

Collaborations:

The collaborations for the Resource Broker pattern are shown in Figure 4.3.

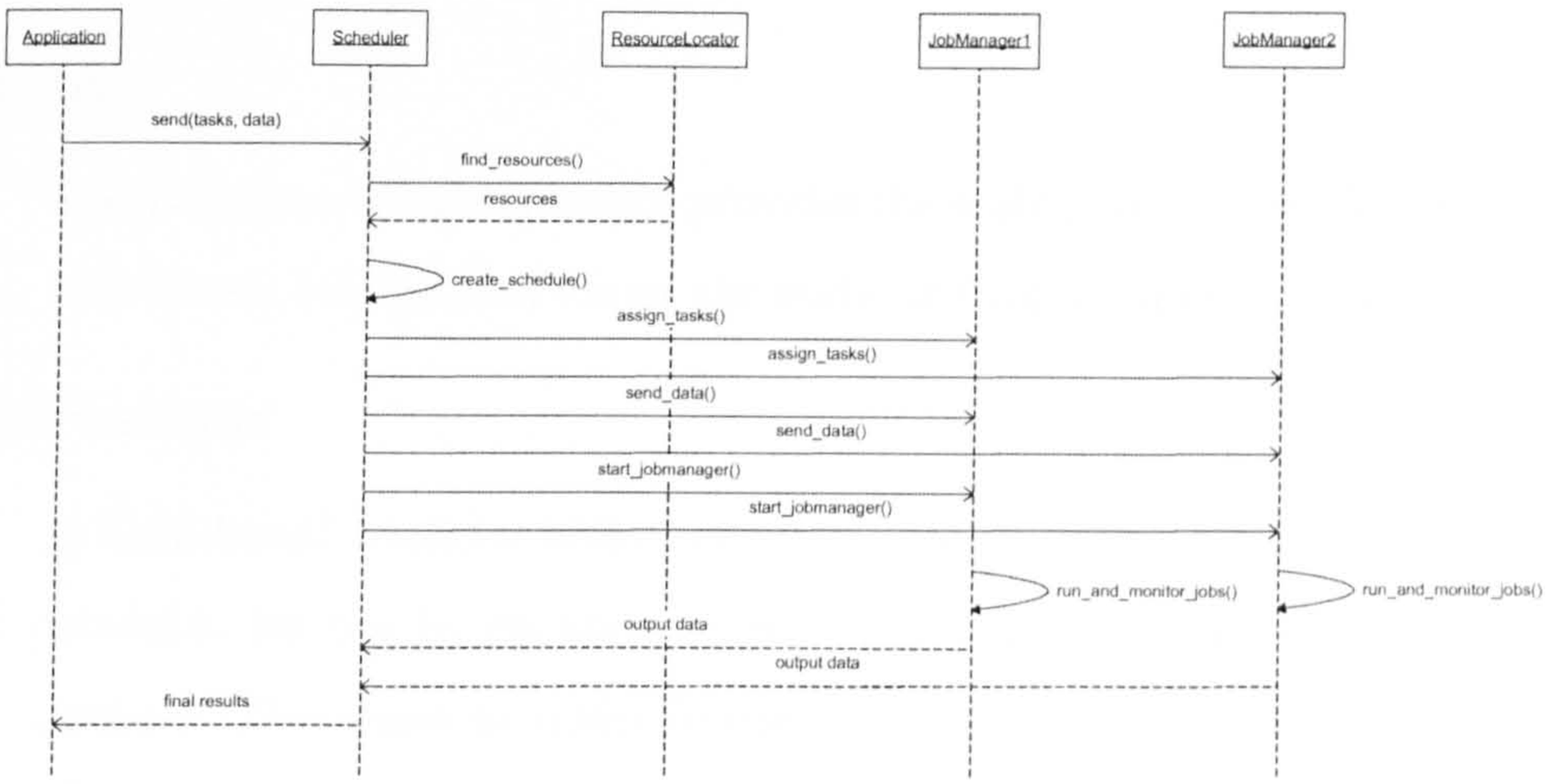


Figure 4.3: A UML Sequence Diagram for the Resource Broker Pattern

Consequences:

The Resource Broker pattern allows an application to discover and use appropriate computational resources. The use of a resource broker allows an application to run tasks on multiple resources and across administrative domains.

Related Patterns:

The Resource Broker pattern relies on the Resource Locator pattern for resource discovery, and the Job Manager pattern for the low-level execution management of individual subtasks.

Resource Locator Pattern

Intent:

The Resource Locator pattern provides the ability to discover Grid resources and obtain information about the state and capabilities of those resources.

Motivation:

A Grid-based resource broker must be able to discover what resources are available for use by an application, as well as the properties of those resources. There may be multiple resource directories containing information about available resources, and these may differ in what information they provide and how they are accessed. The solution is to provide a resource locator component that is capable of interacting with different resource directories and providing the resource broker with a uniform and consistent set of information.

Applicability:

The Resource Locator pattern should be used when:

- you wish to provide a uniform interface to multiple resource directories.
- you wish to provide the resource broker with consistent information about resource properties.

Structure:

The structure of the Resource Locator is shown in Figure 4.4.

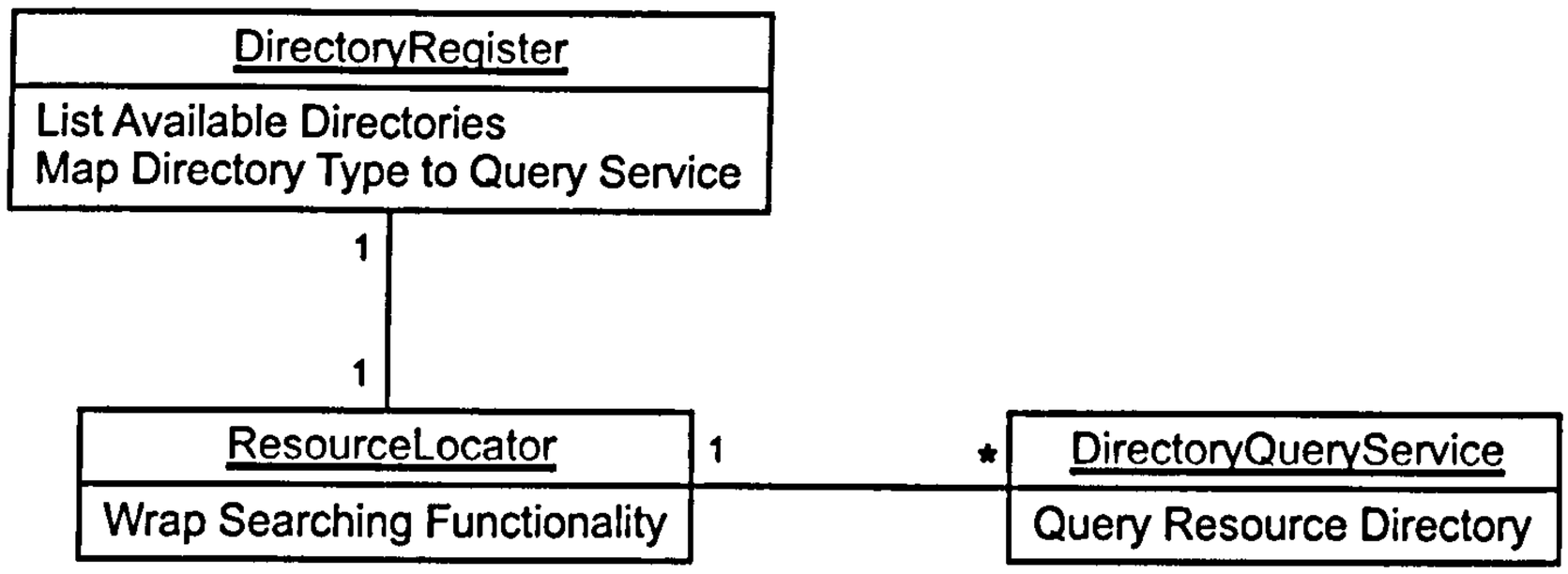


Figure 4.4: A UML Class Diagram for the Resource Locator Pattern

Participants:

- **DirectoryRegistry**
  - provides a list of available resource directories for the ResourceLocator to search, as well as information about how to search them.
- **DirectoryQuery**
  - provides the ResourceLocator with the functionality to search various types of resource directory to find available resources and their properties.
- **ResourceLocator**
  - uses the DirectoryRegistry to get a list of available resource directories to search, and then wraps the functionality of the DirectoryQuery components to provide a uniform way of searching the various resource directories.

Consequences:

The Resource Locator pattern provides a resource broker with a uniform way to search a variety of different resource directories for available resources.

Related Patterns:

This pattern is used by the Resource Broker pattern.

Job Manager Pattern

Intent:

The Job Manager pattern provides low-level control of the execution of application subtasks on a remote machine.

Motivation:

Once a Grid-enabled application has discovered suitable resources, it must then initiate and manage the execution of application subtasks across these resources. Execution management in a heterogeneous Grid computing environment, across multiple administrative domains poses many challenges. One solution is for the application to delegate the responsibility for managing the execution of these subtasks to a job manager agent. This job manager agent can interact with the local resource management systems to initiate and monitor the execution of these subtasks. The job manager agent can also collate the results of these executions and transfer them back to the application.

Applicability:

The Job Manager pattern should be used when:

- the application wishes to initiate the execution of subtasks on a remote resource.
- the complexity of the execution environment makes centralised execution management from the application impractical.

Structure:

The structure of the Job Manager is shown in Figure 4.5.

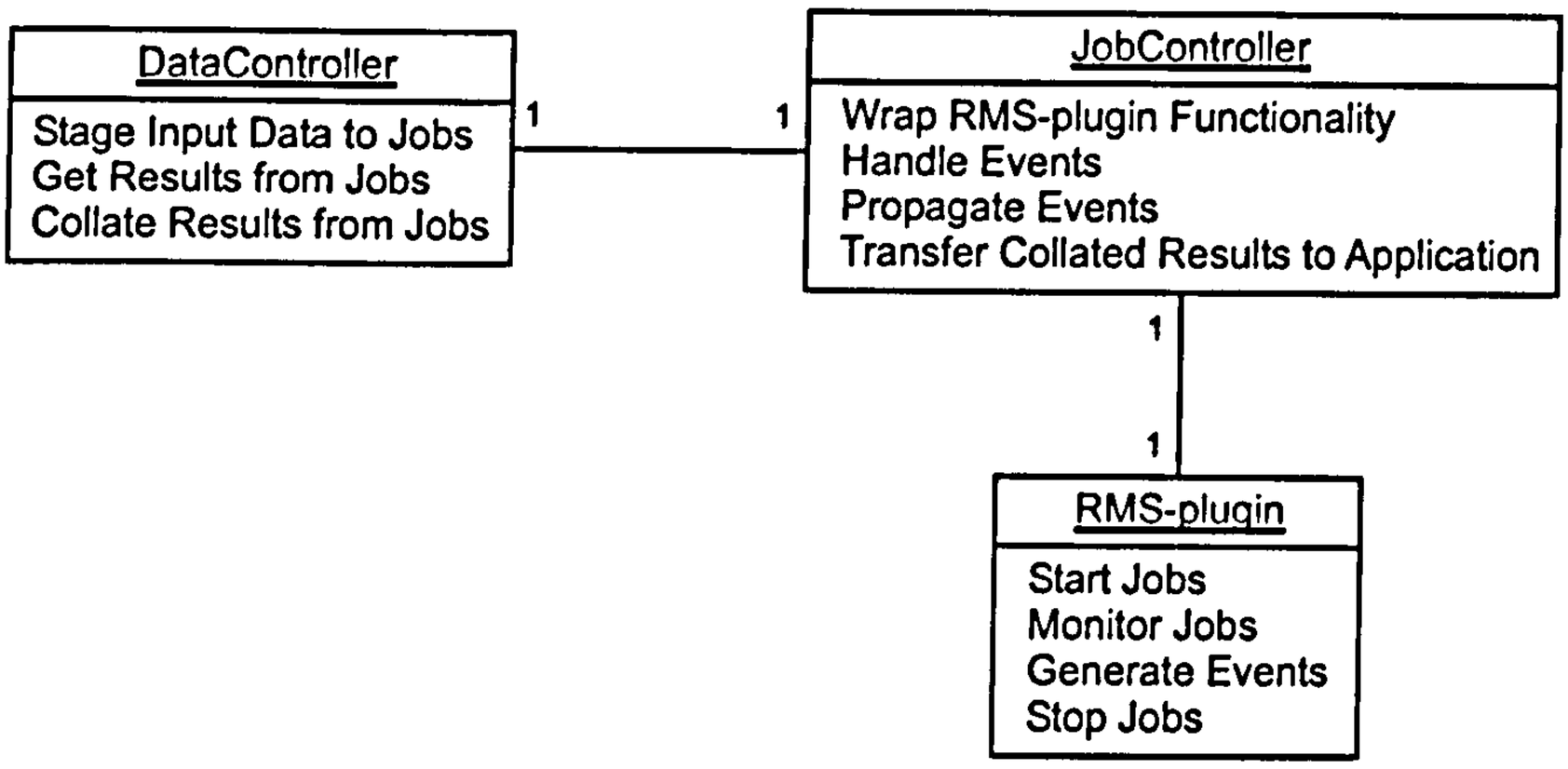


Figure 4.5: A UML Class Diagram for the Job Manager Pattern

Participants:

- **JobController**
  - controls the sequence of execution of jobs.
  - sends the completed results back to the application.
- **DataController**
  - controls data transfer to and from the jobs.
  - collates the results of the application run.
- **RMSplugin**
  - provides a uniform interface to the functionality of the underlying local resource management system.
  - monitors the execution process and performs event notification.

Collaborations:

The collaborations for the Job Manager pattern are shown in Figure 4.6.

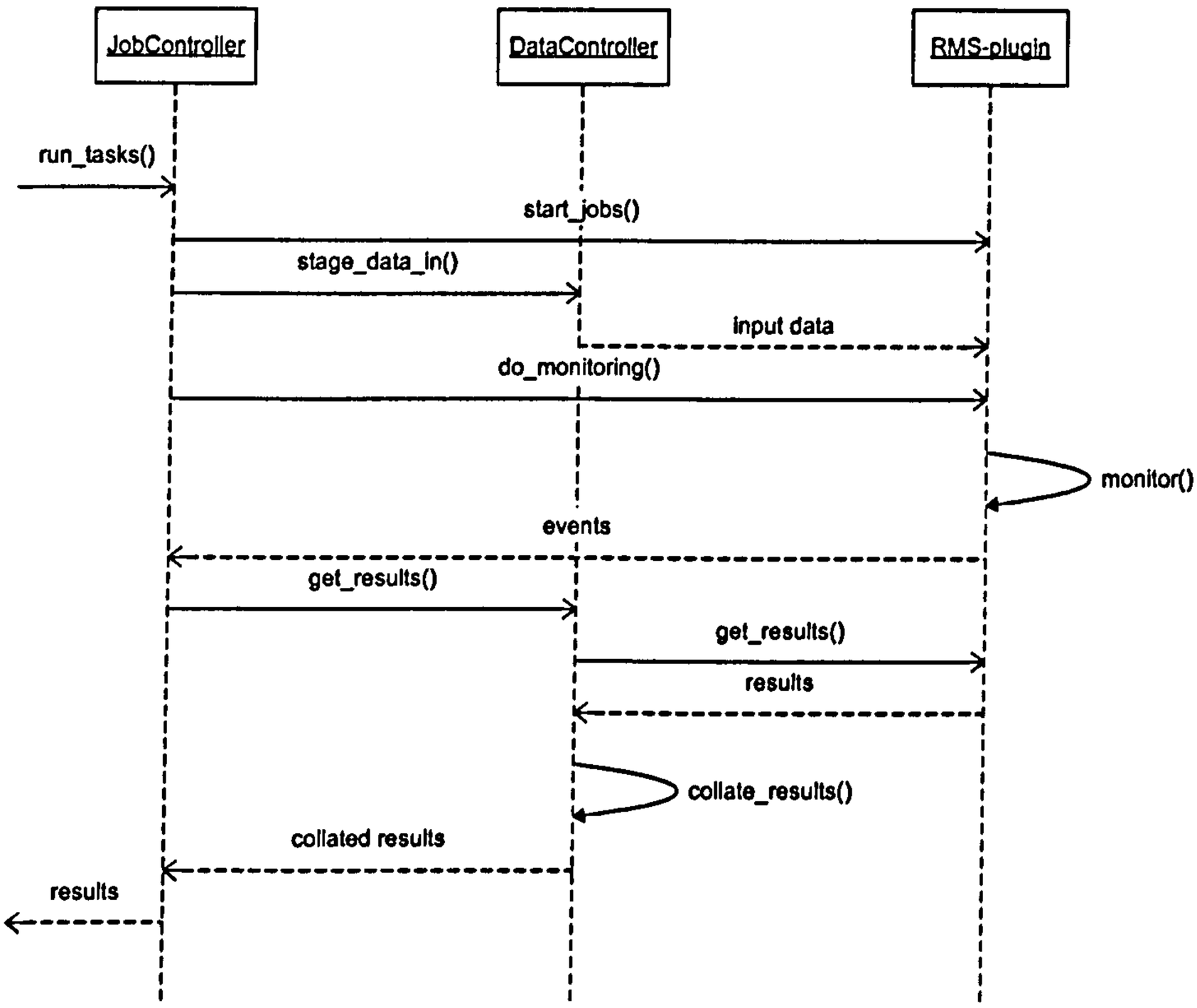


Figure 4.6: A UML Sequence Diagram for the Job Manager Pattern

Consequences:

The Job Manager pattern ensures that the application does not need to interact directly with the remote resources. The job manager agents provide low-level access to Grid resources whilst presenting a uniform interface to the application. Where tight coupling is needed between the application and its subtasks, it may be better to manage the execution directly from the application.

Related Patterns:

This pattern is used by the Resource Broker pattern.

4.5 Workload Allocation Strategies

4.5.1 Introduction

Workload allocation strategies can be classified as either *static* or *dynamic* (Casavant and Kuhl 1988, Wang and Morris 1985, Casey 1981). Static workload allocation schemes use *a priori* information to make scheduling decisions, whilst dynamic workload allocation schemes use runtime information about the state of the system in making decisions. Dynamic schemes often outperform static schemes because they can adapt to changes in workload and system state (Colajanni *et al.* 1998, Banawan and Zeidat 1992). However, dynamic workload allocation schemes have a much higher overhead because they have to collect and process system information at frequent intervals. In a heterogeneous, distributed environment (such as a computational Grid) the collection of this dynamic system information is often time-consuming, unreliable and inconsistent (Schopf 2002). Tang and Chanson (2000) argue that, for some applications, the performance gain from a good low-cost static algorithm may be more beneficial than the overheads involved in collecting detailed dynamic information.

Workload allocation policies for distributed systems are often analysed in terms of queueing theory (see Kleinrock (1975) for an introduction to queueing theory). Tang and Chanson (2000) have investigated the optimisation of static job schedules in networks of heterogeneous[2] workstations. They modelled each computer in their system as an $M/M/1$ queue with arrival rate λ, baseline service rate μ, and relative processing speed s_i. Their aim was to find the optimal

[2]Heterogeneous in the context of this paper refers to differences in computational capability rather than differences in CPU architecture or operating system.

allocation of workload, α_i, for each of the n nodes so as to minimise the mean response time of the system (shown in Equation 4.1 below).

$$\overline{T}_{sys} = \sum_{i=1}^{n} \alpha_i \frac{1}{s_i\mu - \alpha_i\lambda} \tag{4.1}$$

Tang and Chanson (2000) compared the performance of their static workload allocation algorithm to a weighted workload allocation scheme in which workload was allocated to nodes in proportion to their processing speed. They used a discrete event simulator to evaluate the performance of the workload allocation algorithms and showed that their optimised mean response time (ORT) static workload allocation scheme outperformed the weighted workload allocation scheme. This confirmed earlier results by Leslie and McKenzie (1999) who found that, at low and moderate levels of system load, it is more efficient to allocate a disproportionately high share of workload to the more powerful nodes.

He *et al.* (2006) have also used queueing theory to investigate optimal static workload allocations. They have extended the work in Tang and Chanson (2000) to multi-clusters, where each node in the system is a homogeneous multi-cluster consisting of m processing nodes. Each system node is therefore modelled as an $M/M/m$ queue, and two objectives for workload allocation are examined: optimising the mean response time for non-real-time jobs in the system, and optimising the mean miss rate[3] (OMR) for jobs with soft real-time constraints. The performance of the workload allocation schemes developed in He *et al.* (2006) has been evaluated using discrete event simulation, and both optimal strategies perform well with respect to their objectives when compared to a weighted workload allocation scheme.

Banawan and Zeidat (1992) compared two static workload allocation policies with six dynamic policies for several systems under a variety of loads. The

[3]For Quality-of-Service (QoS) demanding jobs, the mean miss rate is the average number of jobs that fail to meet the required QoS level. The profits under a Service Level Agreement (SLA) are maximised if all the jobs meet their QoS demands.

evaluation of the different workload allocation policies was performed using simulation and generally the dynamic policies outperformed the static schemes. The exception to this was when the collection of system information proved to be expensive. In this case Banawan and Zeidat (1992) recommended the use of a static workload allocation policy (such as the one developed in Tang and Chanson (2000) or He *et al.* (2006)).

4.5.2 Modelling the System

Computational Grids such as the White Rose Grid (see Section 4.3) can be modelled as a network of heterogeneous compute resources, $\{r_1, r_2, ..., r_n\}$, with each resource having service rate $\{\mu_1, \mu_2, ..., \mu_n\}$ (see Figure 4.7). The mean arrival rate into the system is λ, and each arrival corresponds to a group of k jobs (many applications, from parameter sweeps to computational fluid dynamics simulations, involve the submission of groups of tasks for execution). The task of the global scheduler in Figure 4.7 is to allocate a fraction of this workload, α_i, to each resource according to the specified scheduling scheme.

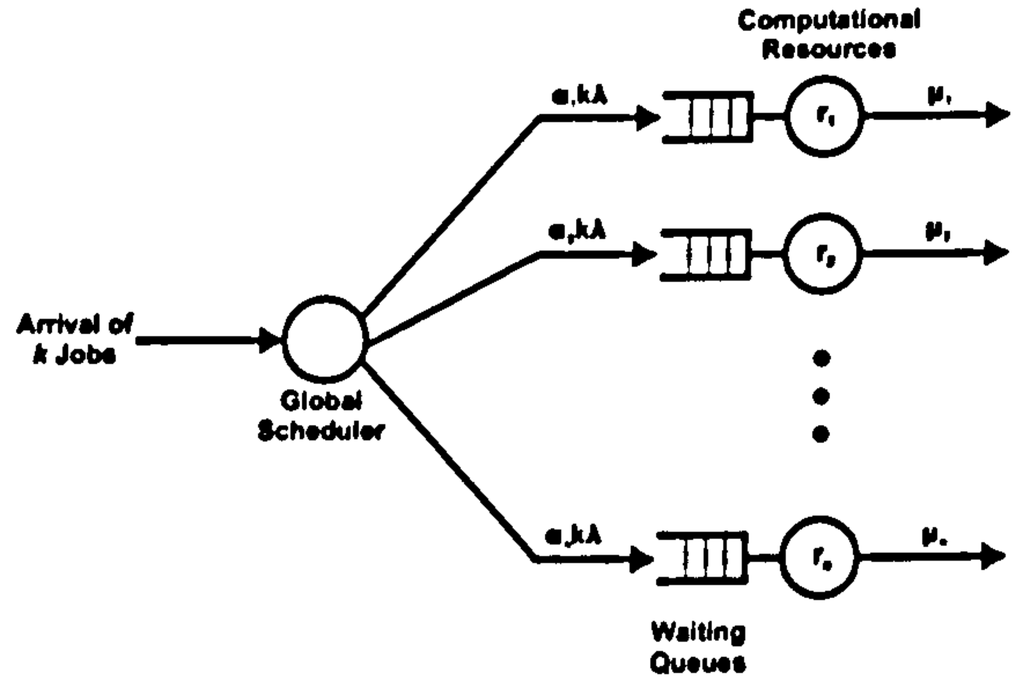


Figure 4.7: The System Model

4.5.3 Workload Allocation Schemes for Bulk Arrivals

Weighted Workload Allocation

A simple workload allocation strategy is to set the fraction of workload allocated to each resource to be proportional to the relative processing speed of that resource, i.e.

$$\alpha_i = \frac{\mu_i}{\sum_{j=1}^{n} \mu_j}$$

This workload allocation strategy takes into account the differences in speeds between resources and attempts to share the workload fairly amongst the resources available. However, Leslie and McKenzie (1999) have shown that this strategy is suboptimal under most system loads. It is frequently used in the literature as a benchmark for comparison against static workload allocation algorithms, since it represents the worst case performance for static workload allocation algorithms (Tang and Chanson 2000, He *et al.* 2006).

Optimal Workload Allocation for Bulk Arrivals

The problem of allocating workload optimally amongst the resources in a system can be expressed as a non-linear optimisation problem using queueing theory, and thus solved mathematically. The goal of the workload allocation strategy discussed here is to minimise the mean response time of jobs in the system[4]. Each resource in Figure 4.7 is modelled as an $M/M/1$ queue, with the mean arrival rate of groups of k jobs into the system given as λ and the service rate of each resource, r_i, given as μ_i.

According to Little's law (Little 1961):

[4]The response time of a job is defined as the time the job spends waiting in the queue plus the time it spends in service.

$$\overline{T} = \frac{\overline{L}}{k\lambda}$$

where $\overline{T}$ = the mean response time of jobs
$\overline{L}$ = the mean number of jobs in a resource
$k\lambda$ = the mean arrival rate of jobs at a resource (4.2)

Using queueing theory, the expression for the mean number of jobs in a resource, $\overline{L}$ is found to be (see Appendix A for the proof):

$$\overline{L} = \frac{k\lambda(k+1)}{2\mu - 2k\lambda} \tag{4.3}$$

Therefore,

$$\overline{T} = \frac{k+1}{2\mu - 2k\lambda} \tag{4.4}$$

The mean response time of resource i $(1 \leq i \leq n)$, $\overline{T}_i$, when allocated a fraction of the k jobs arriving in each batch, $\alpha_i k$, is therefore:

$$\overline{T}_i = \frac{\alpha_i k + 1}{2\mu_i - 2\alpha_i k\lambda} \tag{4.5}$$

The mean response time of the system, $\overline{T}_{sys}$, is:

$$\overline{T}_{sys} = \sum_{i=1}^{n} \alpha_i \cdot \frac{\alpha_i k + 1}{2\mu_i - 2\alpha_i k\lambda} \tag{4.6}$$

The objective of this workload allocation strategy is to find the workload allocation, $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$, that minimises $\overline{T}_{sys}$ (and thus optimises the performance of the system), subject to the constraints[5]:

[5]The last constraint in Equation 4.7 ensures that no resource becomes saturated.

$$\sum_{i=1}^{n} \alpha_i = 1 \quad \text{and} \quad \alpha_i k\lambda < \mu_i \tag{4.7}$$

Rearranging Equation 4.6 (see Appendix B) gives:

$$\overline{T}_{sys} = -\frac{1}{2\lambda} \cdot \sum_{i=1}^{n} \left[\alpha_i - \frac{2\lambda\alpha_i + 2\mu_i\alpha_i}{2\mu_i - 2\lambda\alpha_i k} \right] \tag{4.8}$$

Therefore minimising $\overline{T}_{sys}$ reduces to minimising:

$$F(\alpha) = \sum_{i=1}^{n} \left[\alpha_i - \frac{2\lambda\alpha_i + 2\mu_i\alpha_i}{2\mu_i - 2\lambda\alpha_i k} \right] \tag{4.9}$$

subject to the constraints in Equation 4.7.

This is a constrained minimisation problem and can be solved using Lagrange multipliers (see Bertsekas (1982) for details), giving[6]:

$$\alpha_i = \frac{1}{2\lambda k} \left[2\mu_i - \sqrt{4\mu_i\lambda + 4\mu_i^2} \cdot \frac{\sum_{j=1}^{n} 2\mu_j - 2\lambda k}{\sum_{j=1}^{n} \sqrt{4\mu_j\lambda + 4\mu_j^2}} \right] \tag{4.10}$$

If the service rate of a resource is low, it is possible that the workload fraction, α_i, found by Equation 4.10 may be negative. In this case, the workload allocation, $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$, produced by Equation 4.10 will not be practical, since it is not possible to assign negative workload to a resource. To overcome this problem, α_i (where r_i is the resource with low service rate) should be set to zero, and the workload allocation recomputed without r_i. The pseudocode for calculating the optimal workload allocation for a given set of resources is shown in Algorithm 1.

[6]See Appendix C for the proof

Algorithm 1 Calculate the Optimal Workload Allocation

```
1: procedure OPTIMAL_WORKLOAD_ALLOCATION(k, λ, μ)
2:     Sort μ in ascending order
3:     m = 1
4:     for i = 1, n do
5:         α_i = (1/2λk) [2μ_i − sqrt(4μ_iλ + 4μ_i^2) · (Σ_{j=m}^{n} 2μ_j − 2λk) / (Σ_{j=m}^{n} sqrt(4μ_jλ + 4μ_j^2))]
6:         if α_i < 0 then
7:             α_i = 0
8:             m = m + 1
9:         end if
10:    end for
11:    Unsort α
12: end procedure
```

4.5.4 Performance Evaluation of Workload Allocation Algorithms

Experimental Set-Up

Initially the workload allocation algorithms proposed in Section 4.5.3 were evaluated using discrete event simulation to investigate their static performance under a variety of system configurations and loads. The simulation model consists of multiple compute resources connected by a high-speed network (see Figure 4.7), with data and program files stored on dedicated file servers and thus incurring no communication overhead from file transfer. Incoming jobs arrive at the global scheduler which then distributes them amongst the compute resources according to the workload allocation calculated by the algorithm. The simulation parameters used in this study have been chosen to represent a realistic system.

Each simulation pass was run from the empty system using a stream of 150,000 job batches. The first 50,000 batches are considered the initialisation period so as to allow the system to reach steady state, and are thus discarded. Statistics about the system are therefore collected from the 100,000 batches of jobs that arrive after this initialisation period, and each point in the performance graphs

represents the median of 10 independent runs initialised with different random number seeds.

The performance of the Dynamic Least Load (DLL) workload allocation algorithm (Chanson *et al.* 2000) is also included in the following experiments as it represents an upper bound for the static performance of the proposed algorithms. When a job arrives at the global scheduler the DLL algorithm assigns it to the resource with the lowest instantaneous normalised load, thus ensuring that the resources are loaded as lightly as possible. This algorithm requires the global scheduler to keep track of the load index of each resource.

Effects of Heterogeneity

Heterogeneity in the system model presented in this Chapter is measured by the difference in speed between the fastest and slowest computational resource. The effects of varying this speed differential are shown in Figure 4.8. The system here consists of 10 computational resources, of which 9 are low speed and 1 is high speed. The speed of the slower resources is fixed at 1, whilst the speed of the faster resource is varied between 1 and 20 - thus the system ranges from a homogeneous system to a highly heterogeneous one. In this experiment jobs arrive at the global scheduler in batches of 10, with the arrival rate set at 0.1.

It can be seen from Figure 4.8 that the optimised mean response time workload allocation algorithm significantly outperforms the weighted workload allocation algorithm in heterogeneous systems. As the heterogeneity of the system increases, so the difference in performance between the ORT algorithm and the weighted workload allocation algorithm also increases. When the speed differential between resources is 20:1, the ORT algorithm performs 41% better than the weighted workload allocation algorithm. The reason for this (as shown in Table 4.2) is that, in highly heterogeneous systems, the ORT algorithm assigns a much larger proportion of the workload to the faster resources than the weighted workload allocation algorithm does. This follows a similar strategy to that used by the

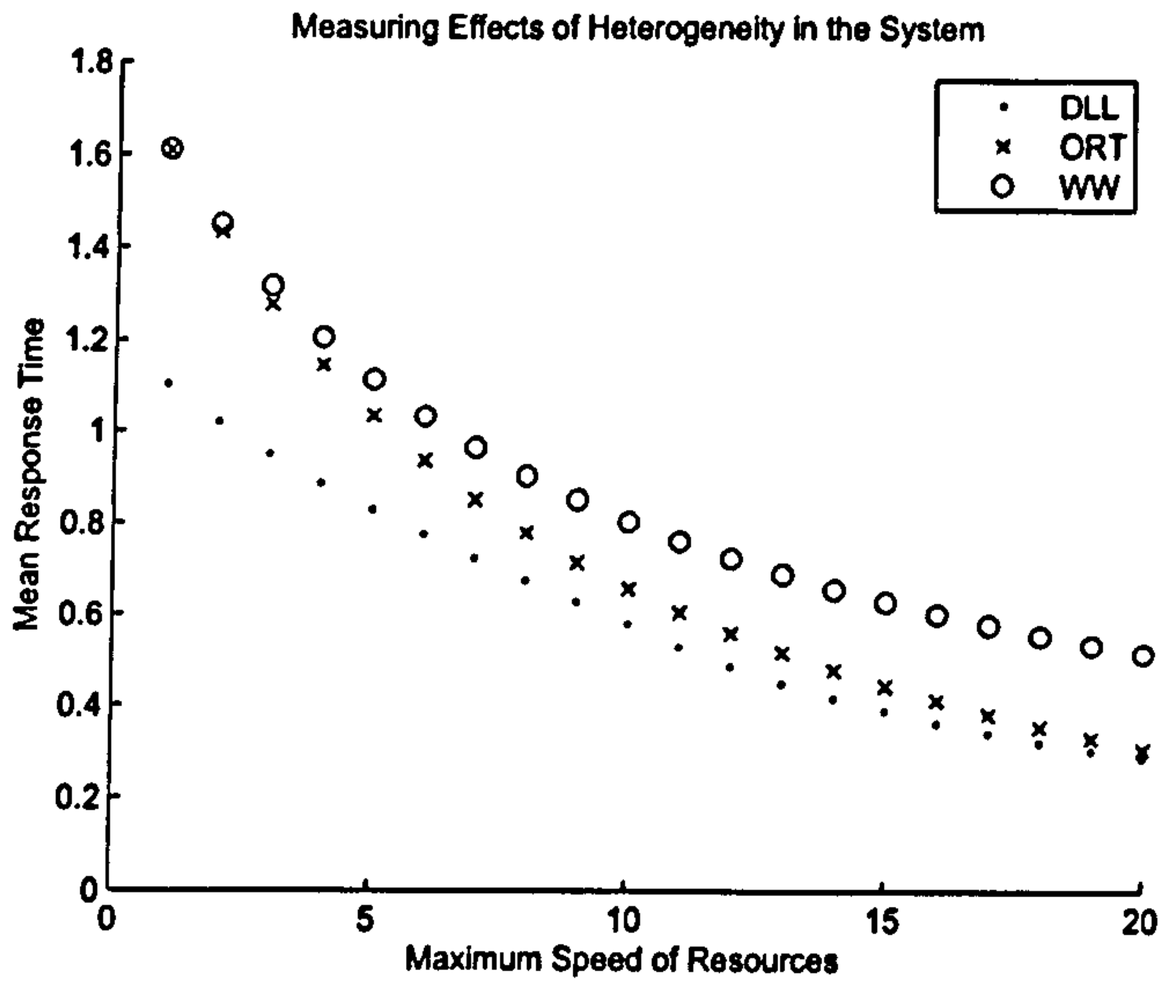


Figure 4.8: Plot of Heterogeneity against Performance

Dynamic Least Load algorithm and is the reason why, as the heterogeneity of the system increases, the performance of the ORT algorithm approaches that of the Dynamic Least Load algorithm (as can be seen in Figure 4.8).

Heterogeneity	Workload Allocated to Fastest Resource		
	DLL	ORT	WW
16:1	0.992	0.888	0.640
17:1	0.996	0.909	0.654
18:1	0.998	0.928	0.667
19:1	0.999	0.946	0.679
20:1	0.999	0.963	0.690

Table 4.2: Workload Allocation in Highly Heterogeneous Systems

Effects of System Size

The scalability of the workload allocation algorithms discussed in this Section is found by measuring their performance when the number of computational resources in the system is varied (shown in Figure 4.9). In this experiment the system consists of an equal number of low speed and high speed resources. The

speed of the slower resources is set to 1, and the speed of the faster resources is set to 10. The number of resources in the system is then varied between 2 and 20. As in the previous experiment, jobs arrive in batches of 10 with the arrival rate set to 0.1.

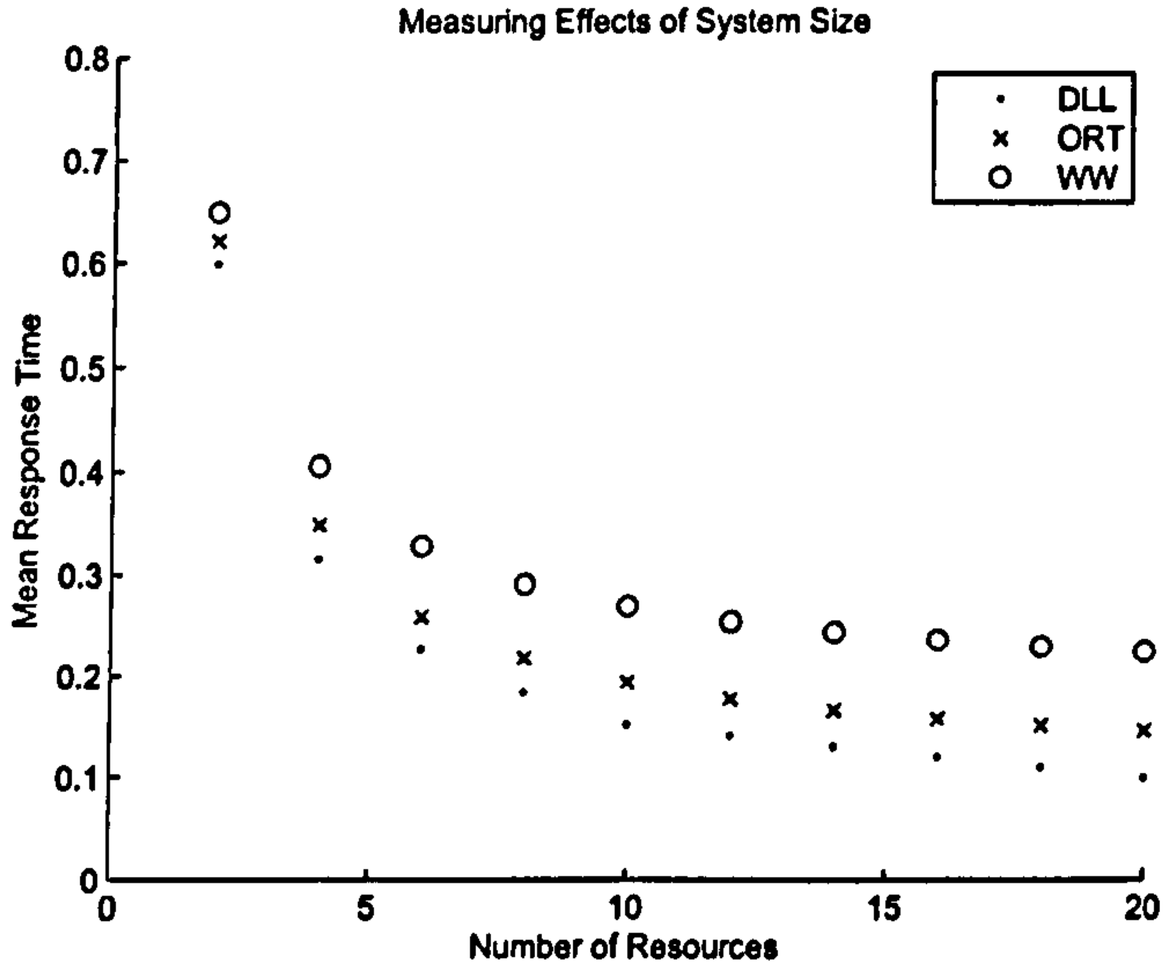


Figure 4.9: Plot of System Size against Performance

Figure 4.9 shows that the ORT workload allocation algorithm performs substantially better than the weighted workload allocation algorithm, and that the difference in performance between the two algorithms increases as the size of the system grows. When there are 6 resources in the system, the ORT algorithm outperforms the weighted workload allocation algorithm by 21%. However, when there are 20 resources in the system, this difference in performance increases to 35%. The reason for this difference is that, as explained previously, the ORT workload allocation algorithm assigns a disproportionately large fraction of the workload to the fastest resources.

It can likewise be seen from Figure 4.9 that the difference in performance between the optimised mean response time algorithm and the Dynamic Least Load algorithm also increases as the system grows. This is because the

ORT workload allocation algorithm assigns equal fractions of workload to resources that have equal speeds, whilst the Dynamic Least Load algorithm uses instantaneous load information to dynamically balance the workload between resources (Tang and Chanson 2000).

Table 4.3 shows the difference in the average workload fractions assigned by both algorithms when the system consists of 10 resources. As can be seen in Table 4.3, the Dynamic Least Load algorithm assigns different fractions of the workload to resources that have the same speed based on the instantaneous capacity of that resource.

Speed of Resource	Workload Allocated to Resource	
	DLL	ORT
0.0	**0**	**0**
10.0	0.199	0.167
10.0	0.199	0.167
10.0	0.198	0.167
10.0	0.198	0.167
10.0	0.103	0.167
10.0	0.102	0.167

Table 4.3: Workload Allocation when the System has 12 Resources

Effects of Batch Size

As discussed in Section 4.5.2, jobs arrive at the global scheduler in batches. Figure 4.10 shows the impact that the number of jobs arriving in each batch[7] has on the performance of the workload allocation algorithms. The system under consideration in this experiment consists of 10 computational resources with varying speeds (see Table 4.4 for the configuration of the system). Batches of jobs arrive at the global scheduler in the system with arrival rate equal to 0.1.

It can be seen from Figure 4.10 that the optimised mean response time

[7]The size of the batches of jobs is the primary factor governing the level of workload in the system; the smaller the batch size, the lower the workload level.

Speed of Resource	Number of Resources
1.0	4
2.0	3
5.0	2
10.0	1

Table 4.4: System Configuration

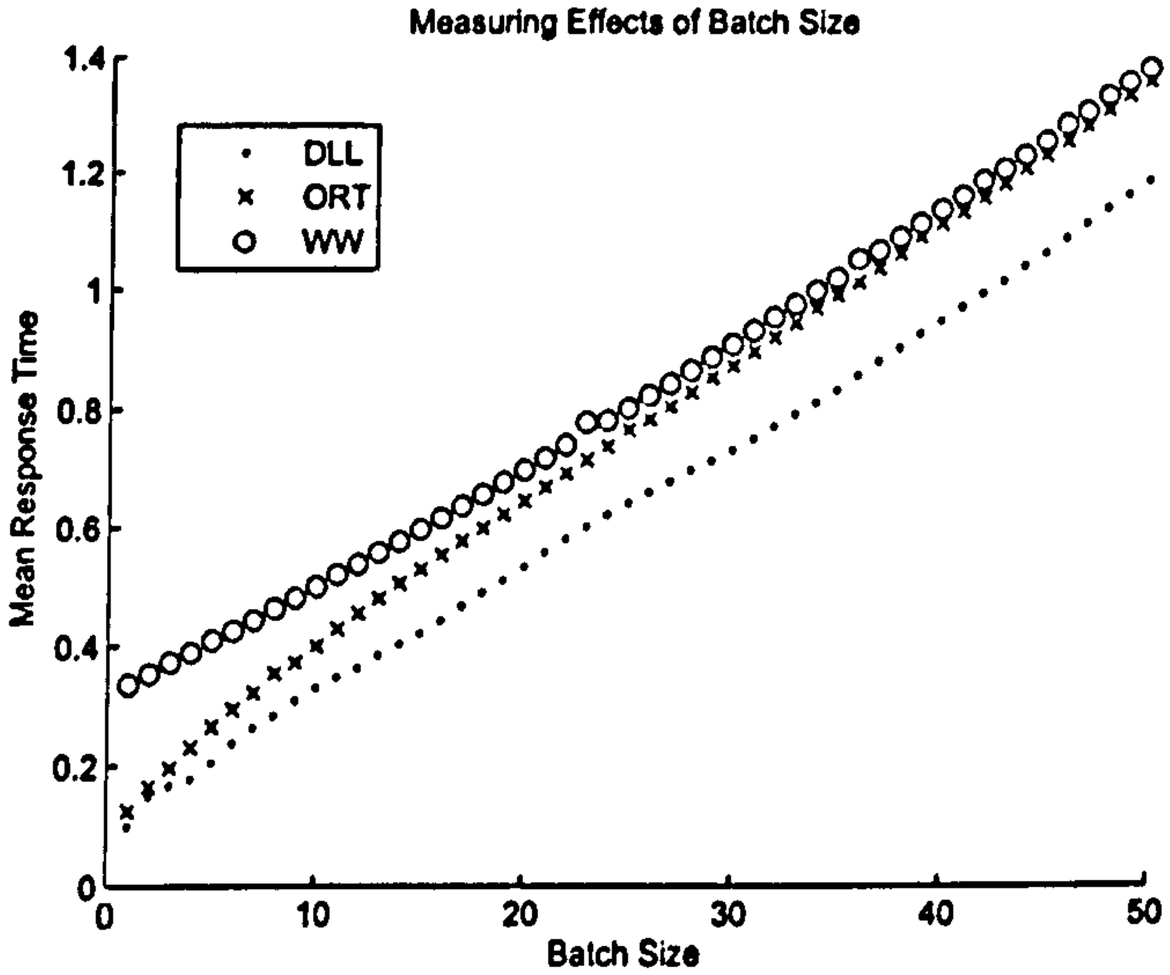


Figure 4.10: Plot of Batch Size against Performance

workload allocation algorithm performs significantly better than the weighted workload allocation algorithm when the number of jobs arriving in each batch is low. However, it can also be seen that, as the batch size increases, the performance of the ORT algorithm tends to that of the weighted workload allocation algorithm. When jobs arrive individually at the global scheduler the ORT algorithm performs 62% better than the weighted workload allocation algorithm. This difference in performance is reduced to 20% when there are 10 jobs arriving in each batch, and 1.5% when there are 50 jobs in each batch.

The reason for this trend in performance is that the weighted workload allocation algorithm assigns the same fraction of the workload to each resource regardless of the load on the system; whereas, under low system loads, the ORT algorithm assigns a disproportionately large fraction of the workload to the faster resources (as explained previously). Since jobs spend less time waiting to be run when the system load is low, assigning a higher proportion of the workload to the faster resources results in smaller response times than if commensurate fractions of the workload were allocated to each resource because the faster resources can process more 'work'. However, under higher system loads it is more beneficial to allocate workload in proportion to the speeds of the resources, and thus the ORT algorithm behaves more like the weighted workload allocation strategy.

Figure 4.10 also shows that the difference in performance between the optimised mean response time workload allocation algorithm and the Dynamic Least Load algorithm, although small when the system is lightly loaded, increases as the load on the system grows. This is because balancing the load dynamically ensures that the best use is made of the available resources in any given time period. Under heavier system loads this becomes more important because the time that jobs spend waiting for service is more significant. Figure 4.11 shows that these performance trends continue when the batch size is very large.

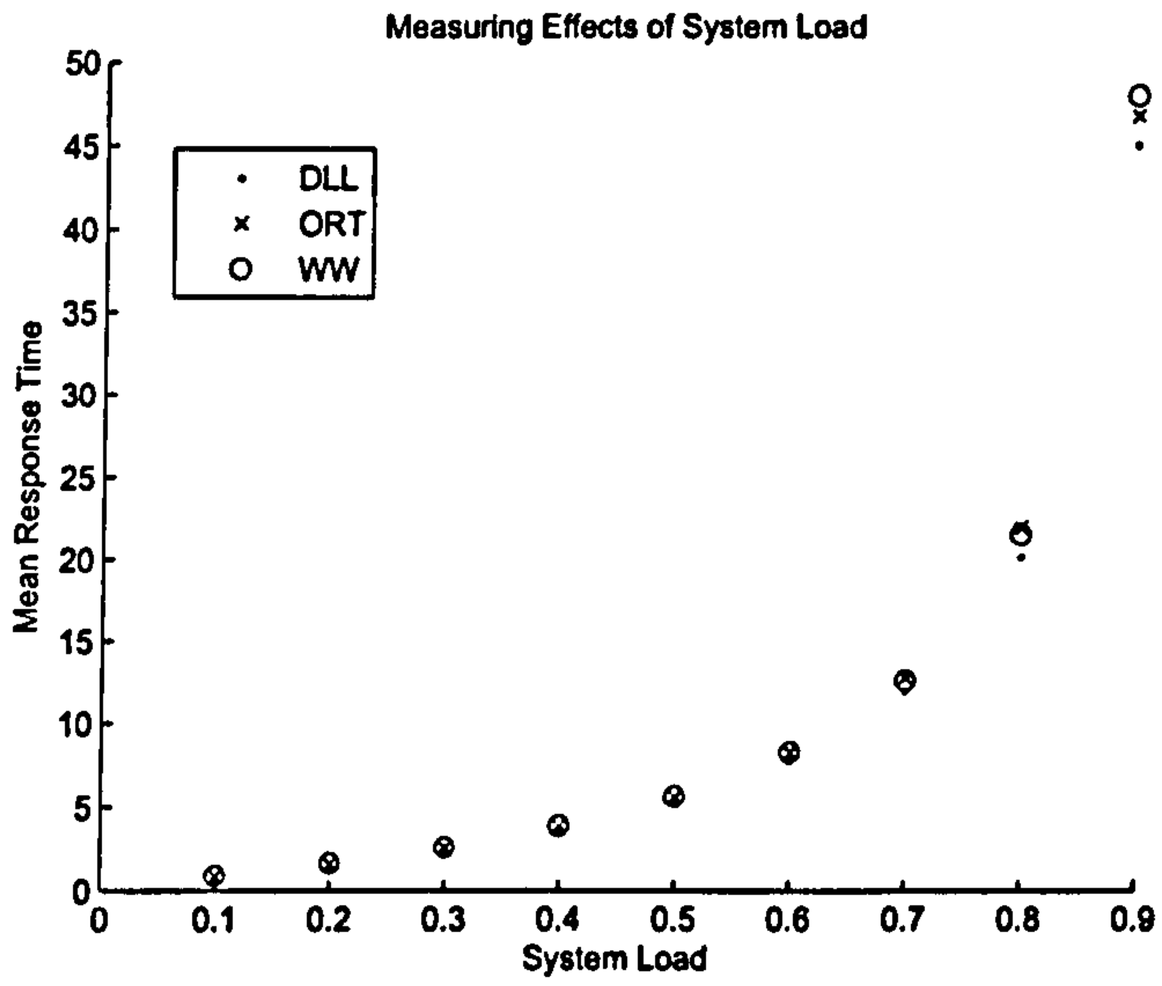


Figure 4.11: Plot of System Load against Performance

Comparing the Optimal Workload Allocation Algorithm for Bulk Arrivals with the Optimal Workload Allocation Algorithm for Jobs Arriving Individually

Following the static performance evaluation of the proposed workload allocation algorithms, a comparison between the ORT algorithm for bulk arrivals proposed here and the ORT algorithm for jobs arriving individually (as proposed in Tang and Chanson (2000)) was undertaken. The configuration of the system was the same as in Table 4.4, and batches of jobs arrive at the global scheduler with arrival rate equal to 0.1.

It can be seen from Figure 4.12 that, although the performance of the two algorithms is similar for batches of less than 5 jobs, the ORT algorithm for bulk arrivals of jobs quickly outperforms the ORT algorithm for single job arrivals as the size of the batch increases. This performance difference is due to the ORT workload allocation algorithm for single job arrivals allocating a larger fraction of workload to the fastest resource than it can process effectively. This happens because the ORT algorithm for jobs arriving individually assumes that job

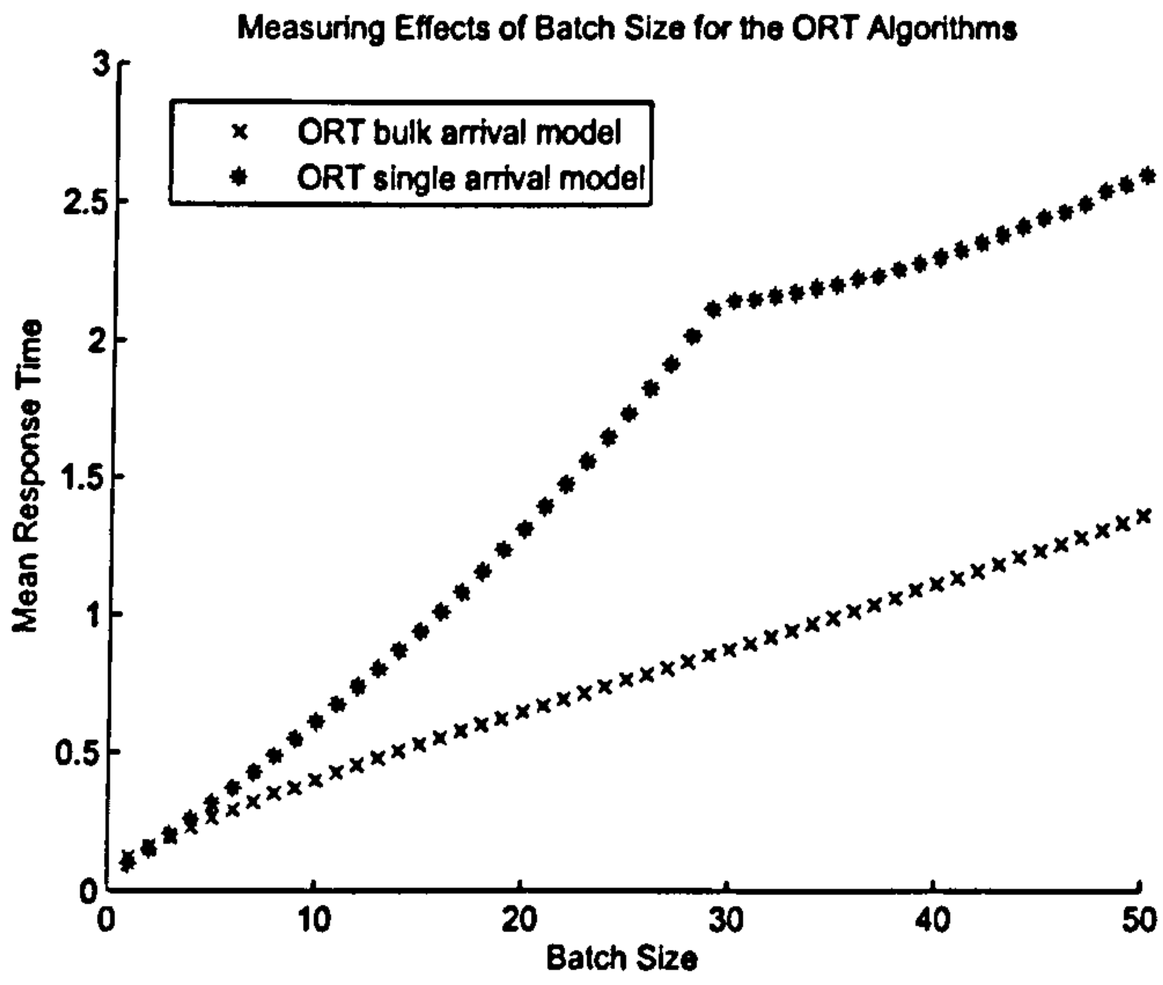


Figure 4.12: Comparison of the ORT Algorithm for Bulk Arrivals with the ORT Algorithm for Individually Arriving Jobs

arrivals will be exponentially distributed and, since jobs are arriving together, this assumption no longer holds - leading to the fastest resource becoming overloaded.

Up until the number of jobs in each batch is equal to 30, the ORT algorithm for single job arrivals allocates all the workload to the fastest resource. After this point the workload is also shared amongst the two resources with service rate equal to 5.0, although the algorithm still overloads the fastest resource.

4.6 Introducing a Novel Hybrid Approach to Job Scheduling

In this Section a hybrid approach to job scheduling is proposed that combines the low cost of a static scheme with the adaptiveness and efficiency of dynamic strategies. Instead of the difficult and computationally expensive task of collecting frequent detailed information about the state of the system, the proposed hybrid scheduling strategy uses information about the response time of

previous jobs through the system to estimate the relative speeds of the compute resources available. It then uses these estimates to schedule the next batch of jobs.

Following the evaluation of the static performance of the proposed workload allocation algorithms using discrete event simulation, a real-world comparison was undertaken between the performance of the hybrid job scheduling approach proposed here and the performance of the static approach used in Tang and Chanson (2000) and He *et al.* (2006). This comparison is described below, and results from a live production Grid computing environment are presented.

Static Job Scheduling Versus a Hybrid Approach

After the static performance of the proposed workload allocation algorithms was evaluated using discrete event simulation, a comparison between the hybrid job scheduling approach proposed in this Chapter and the static approach widely used in the literature was performed. These two approaches to job scheduling were evaluated using the White Rose Grid (see Section 4.3), a production quality Grid computing environment with compute resources located at multiple, geographically distributed sites.

Each run of the experiment consisted of both static and hybrid approaches to job scheduling independently allocating 100 sequential batches of jobs amongst the available resources in the Grid. The size of each batch was chosen to be 20 jobs, so as to represent a typical real-world application such as a grid search or particle swarm optimisation, and 10 independent runs of the experiment were performed at different times of day[8]. The mean arrival rate of batches of jobs at the scheduler and the mean service rates of the Grid resources used to calculate the static workload allocation were found by observing the system over a warm-

[8]This was done to investigate the performance of the two approaches under a variety of system loads. The load on the system was typically lowest in the early morning or at the weekend.

up period immediately prior to the calculation of the workload allocation, whilst the hybrid scheduler was set up so that each resource would be sent the same workload initially (this explains the poor performance for the first iteration of the hybrid job scheduling approach shown in Figure 4.13).

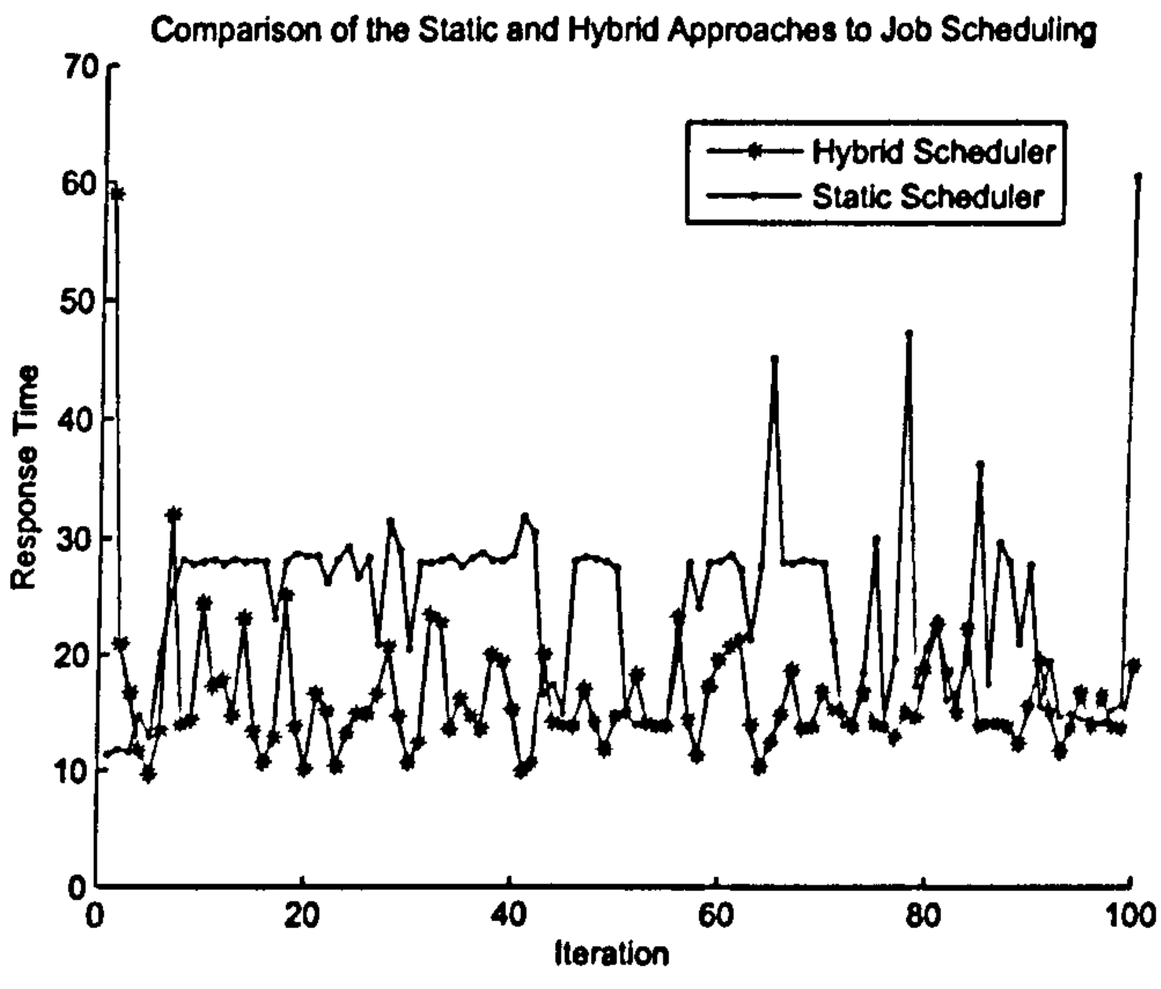


Figure 4.13: Comparison of the Performance of the Hybrid and Static Approaches to Job Scheduling

Figure 4.13 shows that the static scheduling approach initially performs well, with its performance over the first 5 iterations comparable to that of the hybrid approach. However, as the experiment progresses it can be seen that the proposed hybrid approach outperforms the static scheduler. This is because, whilst the workload allocated using the static approach is optimal with respect to the conditions at the start of the experiment, as those conditions change so the optimal workload allocation also changes. Since the workload allocation is fixed when using the static approach, the workload allocated amongst the Grid resources will become sub-optimal as the condition of the system changes[9].

[9]For example, if the workload allocation under the conditions at the start of the experiment was {5, 2, 3} and the performance of the first resource deteriorates, then the static scheduler will still assign 5 jobs to that resource - even though the performance of the system may be better if more jobs were assigned to resources 2 and 3.

Figure 4.14 shows that the proposed hybrid job scheduling approach alters the workload allocation based on the previous response times of the Grid resources. In Figure 4.15 it can be seen that, in response to the increased response time of the system in iteration 56, the workload allocation for iteration 57 is altered. This increase in response time was due to a deterioration in the performance of the Grid resource based at the University of York, and therefore the hybrid scheduler reduced the fraction of the workload assigned to that resource and reallocated it amongst the other available resources in the Grid. As can be seen, this results in an improvement in the performance of the system.

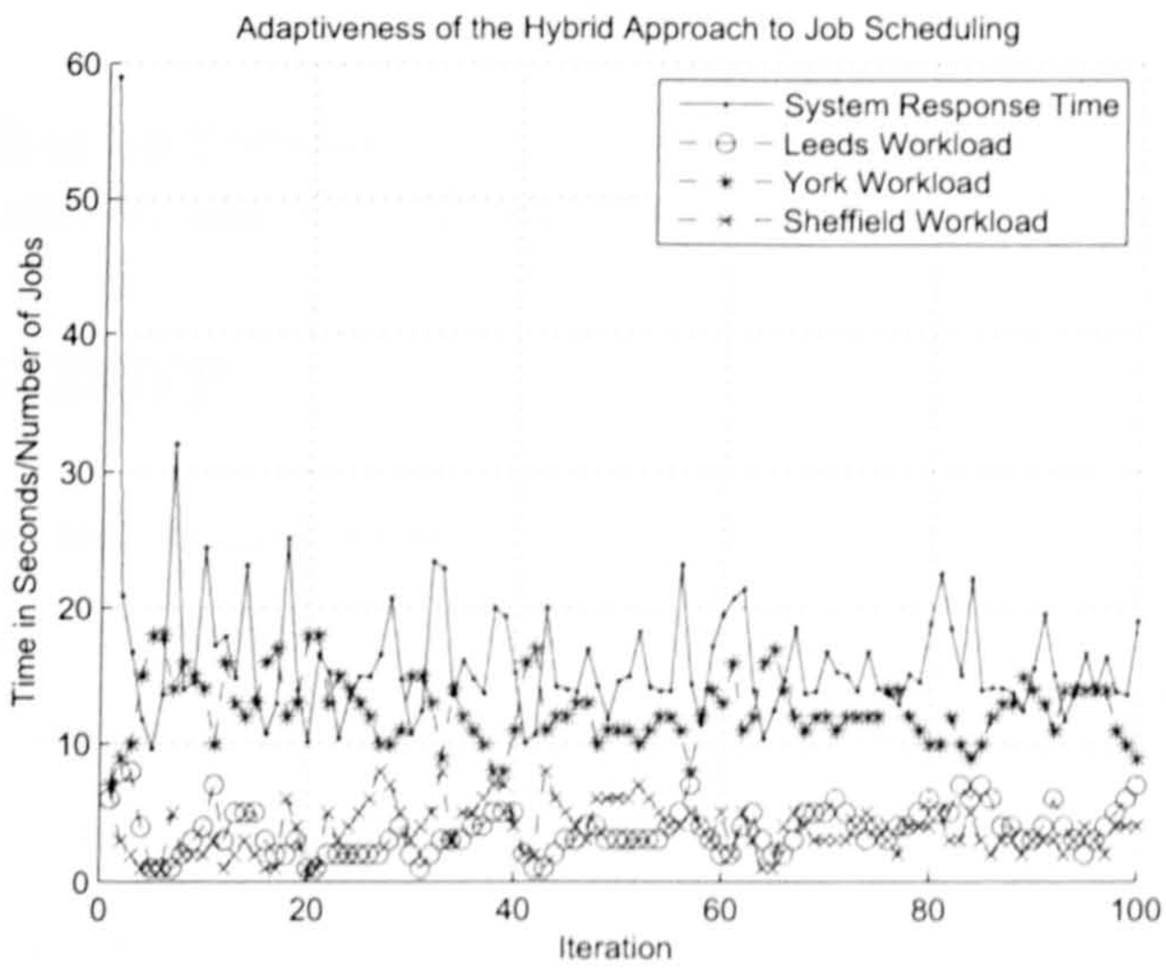


Figure 4.14: Illustration of the Adaptiveness of the Hybrid Job Scheduling Approach

The results shown in Figures 4.13, 4.14, and 4.15 are from a single representative experimental run; however, the hybrid approach to job scheduling proposed in this Chapter performed on average 14.3% better than the static approach over all the runs of the experiment.

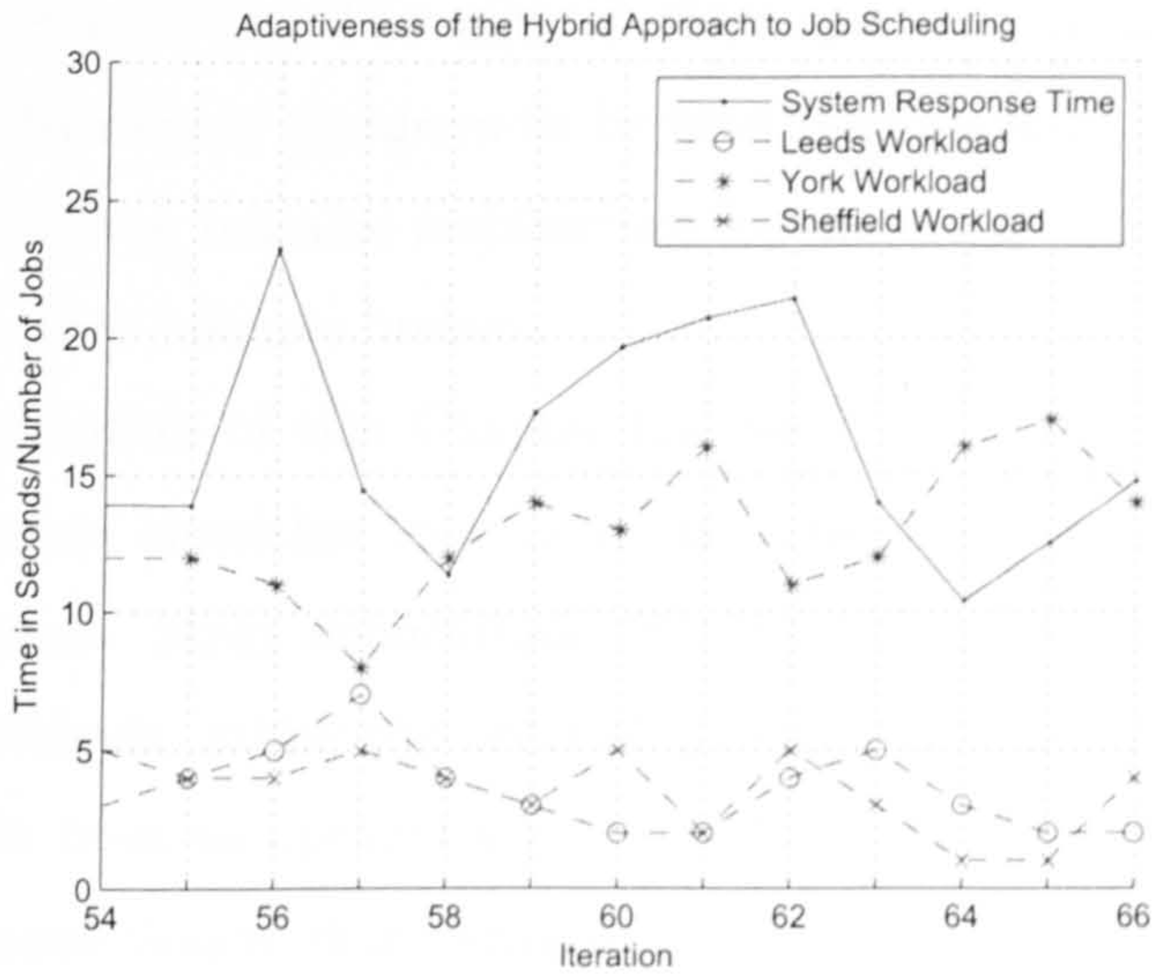


Figure 4.15: Close-up Showing the Response of the Scheduler to a Degradation of the Performance of one of the Resources

4.7 Summary

The distribution and management of computational tasks across a diverse set of geographically distributed heterogeneous resources that are not under centralised control is a critical issue in the realisation of true computational Grids. Many established resource management systems exist to address the problems of scheduling at a local level, and a survey of the most widely used of these was undertaken in Section 4.2.1, but the dynamic and decentralised nature of Grid computing environments introduces new challenges not addressed by these local RMSs. More recently meta-scheduling architectures have been proposed to tackle the key issue of decentralised control over Grid resources by providing a common way of interacting with a variety of resource management systems.

This Chapter has presented a novel set of design patterns to facilitate Grid-based application developers in constructing customised meta-schedulers. These design patterns are the result of practical experience gained in implementing a meta-scheduling architecture for use in a real-world production quality Grid

computing environment. The aim of these patterns is to allow multiple heterogeneous distributed resources to be used transparently by an application by providing a proven reusable architecture for the construction of application-centric Grid-enabled resource brokers.

A key contribution of this Chapter has been the development of a novel workload allocation algorithm to optimise the response time of batches of jobs through the system. Many applications, from parameter sweeps to evolutionary optimisation methods, utilise the processing of multiple independent tasks and can thus benefit from an optimal workload allocation algorithm for batches of jobs. Previous analytical work in deriving optimal workload allocation algorithms has concentrated on jobs arriving individually (Tang and Chanson 2000, He *et al.* 2006) and, as can be seen in Section 4.5.4, these algorithms are not optimal when multiple jobs arrive together. This Chapter has also shown that the novel workload allocation algorithm proposed here exhibits better static performance than the weighted workload algorithm explained earlier (a common benchmark in the literature).

Another major contribution of this Chapter was the proposal of a novel hybrid approach to job scheduling that combines the low computational cost of a static scheduling policy with the adaptive capabilities of dynamic methods. This approach uses response time information from previously completed jobs to estimate the current computational capacity of the available resources, and thus make decisions about where to schedule jobs. The novel job scheduling approach proposed in this Chapter overcomes the need to explicitly query resources for their status (a process that can be both complex and expensive), and instead uses information inherent in the previously completed tasks.

Chapter 5

Evolutionary Optimisation Using Computational Grids

5.1 Introduction

Soft computing techniques such as neural networks, fuzzy logic, and evolutionary computation have been used to solve many real-world problems in engineering design. These techniques can provide the engineer with a powerful set of tools that often outperform conventional methods, especially in areas where the problem domain is noisy or ill-defined. However, in the cases of neural networks and evolutionary computation especially, these tools can be computationally expensive to use.

Grid computing, coupled with the development of parallel algorithms, offers a potential solution to the computationally expensive nature of these techniques. The Grid computing paradigm (see Chapter 3) is an emerging field of computer science that aims to offer "a seamless, integrated computational and collaborative environment" (Baker *et al.* 2002). Ian Foster defines a computational Grid as "a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" (Foster

and Kesselman 1999). Grid computing is differentiated from conventional distributed computing by its emphasis on coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organisations (Foster *et al.* 2001). These resources can include software packages, compute resources, sensor arrays, data and many others.

This Chapter examines the use of evolutionary optimisation within Grid computing environments to assist in solving complex engineering design problems. A key decision in parallelising evolutionary optimisation methods is in which model of parallelisation to use. Section 5.2 provides an overview of strategies that have previously been used in developing parallel algorithms, whilst Section 5.3 compares the two most commonly used models for parallel evolutionary computation. The Grid-enabled optimisation framework developed as part of this research is introduced in Section 5.4, and it is then used to solve two computational expensive real-world engineering design problems in Section 5.5. Section 5.6 summarises the results obtained in this Chapter and presents some guidelines for the use of the proposed Grid-enabled optimisation framework.

5.2 Parallel Evolutionary Optimisation

5.2.1 Introduction to Parallel Evolutionary Algorithms and Population Topologies

The computationally expensive nature of the evaluation process of evolutionary algorithms has motivated the development of parallel EAs. Early approaches to the implementation of parallel evolutionary algorithms can be classified into two categories which still apply today: single-population global EA implementations and EA implementations with multiple communicating populations (Cantú-Paz and Goldberg 1999).

Single-population parallel evolutionary algorithms consist of a single *panmictic*

population maintained globally. This form of parallelism may be effectively exploited using the well established Master-Worker paradigm from parallel computing (see Figure 5.1). Typically the evaluation of candidate solutions in the algorithm is distributed amongst the worker nodes whilst the master node applies the evolutionary operators, such as selection and variation, centrally to the whole population (Fogarty and Huang 1991). Chipperfield and Fleming (1995) also describe a similar scheme where both the evaluation of candidate solutions and the variation operators are performed by the worker nodes.

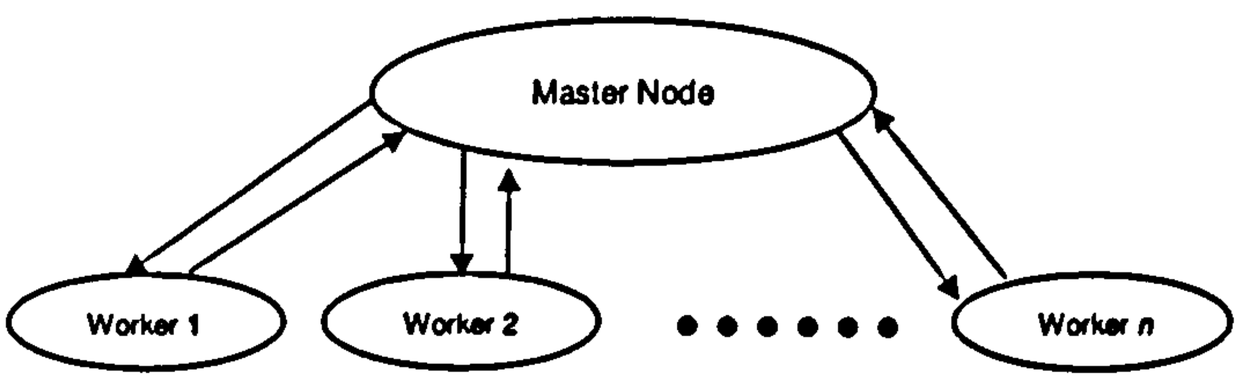


Figure 5.1: The Master-Worker Paradigm

These single-population, globally parallel EAs represent an important case of parallelism because they are functionally equivalent to serial EAs. This means that existing EA theory and design guidelines can easily be applied to their use (Cantú-Paz and Goldberg 1999). Although this type of strategy does not exploit all the parallelism inherent in the evolutionary algorithm, substantial improvements in performance can be achieved - especially in cases where the evaluation of candidate solutions is significantly more computationally expensive that the evolutionary operators themselves (Chipperfield and Fleming 1995).

Evolutionary algorithms with multiple communicating populations can be further divided into those that implement a *coarse-grained* parallelism and those that implement a *fine-grained* parallelism. Algorithms that implement a coarse-grained parallelism (also known as *island* or *migration* EAs) introduce a degree of geographical isolation into the search. The population is divided up into multiple subpopulations (known as *demes*), with each subpopulation evolving

independently (Chipperfield and Fleming 1995). Periodically migration occurs to allow an exchange of information between subpopulations (Rivera 2001). Figure 5.2 shows an example of a coarse-grained island EA using a ring topology, although it should be noted that other network topologies and interconnections are equally applicable.

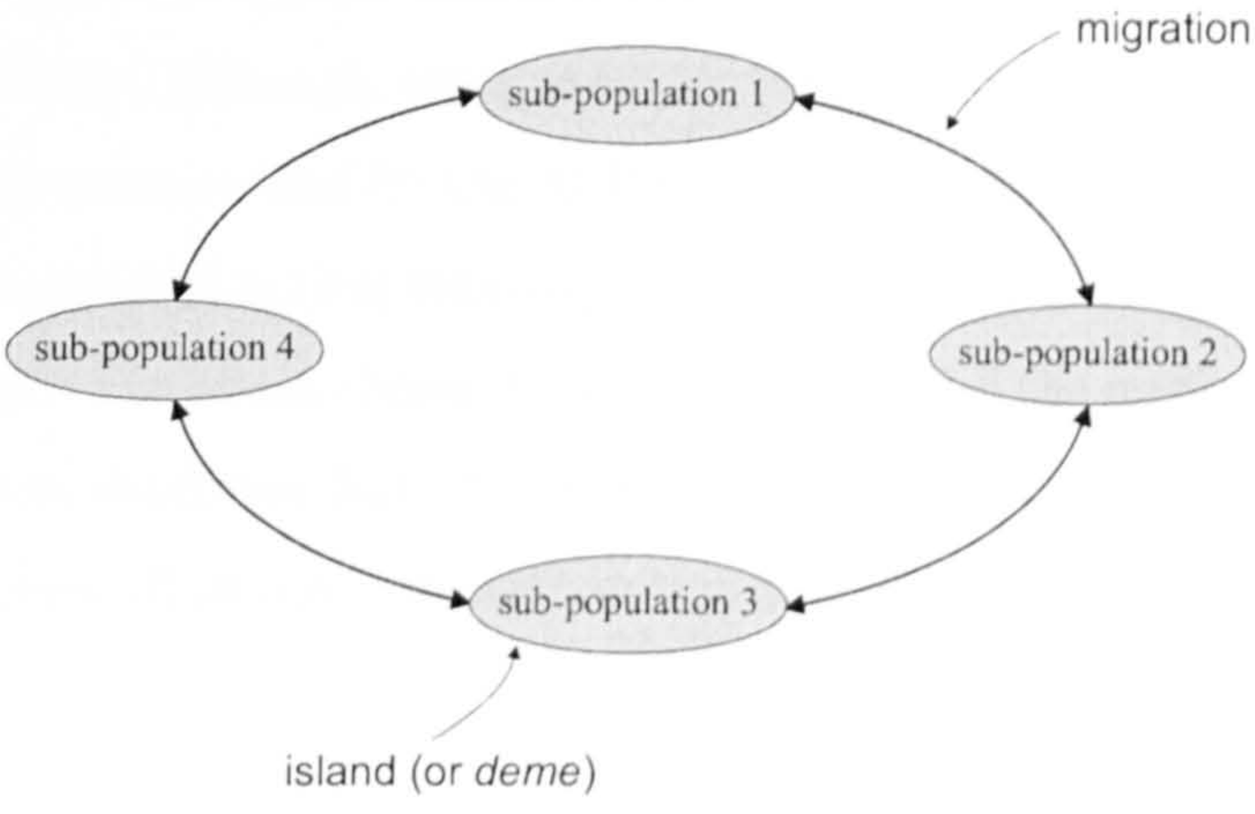


Figure 5.2: A Coarse-Grained Island Evolutionary Algorithm in a Ring Topology

Grosso (1985) is credited with the introduction of the first island EA, although it was implemented sequentially rather than in a parallel architecture. Grosso (1985) divided the population into 5 demes and studied the effects of different rates of migration. The results of this study showed that, for migration rates in a certain critical range, the semi-isolated nature of the subpopulations improved the convergence of the EA by allowing different demes to explore different areas of the search space. Similar results were also reported by Tanese (1987) and Starkweather *et al.* (1991).

Fine-grained parallel EAs (also known as *diffusion* EAs) treat the population as a single continuous structure (Chipperfield and Fleming 1995). In these diffusion EAs a grid is formed to cover the population surface, and each member of the population is assigned to a node in that grid (with each node ideally hosted on a separate processor). The evolutionary operators are then applied to individuals

in the same local neighbourhood (usually chosen to be the adjacent nodes). Rivera (2001) notes that the topology of the network in diffusion EAs strongly determines the behaviour of the algorithm. Diffusion EAs are particularly suited to implementation on symmetric multiprocessor machines (see Section 3.2.2) due to their tightly coupled nature.

Figure 5.3 shows a typical diffusion EA with 25 candidate solutions arranged in a grid topology (although other network topologies are equally possible; for example, the processors used by the ASPARAGOS parallel EA (Gorges-Schleuter 1989) are connected in a ring topology). The candidate solution highlighted in black in Figure 5.3 has been chosen for reproduction, and the mating pool consists of the candidate solutions from the neighbouring nodes (shown in grey in the Figure). The new child solution will replace the parent solution in the marked node.

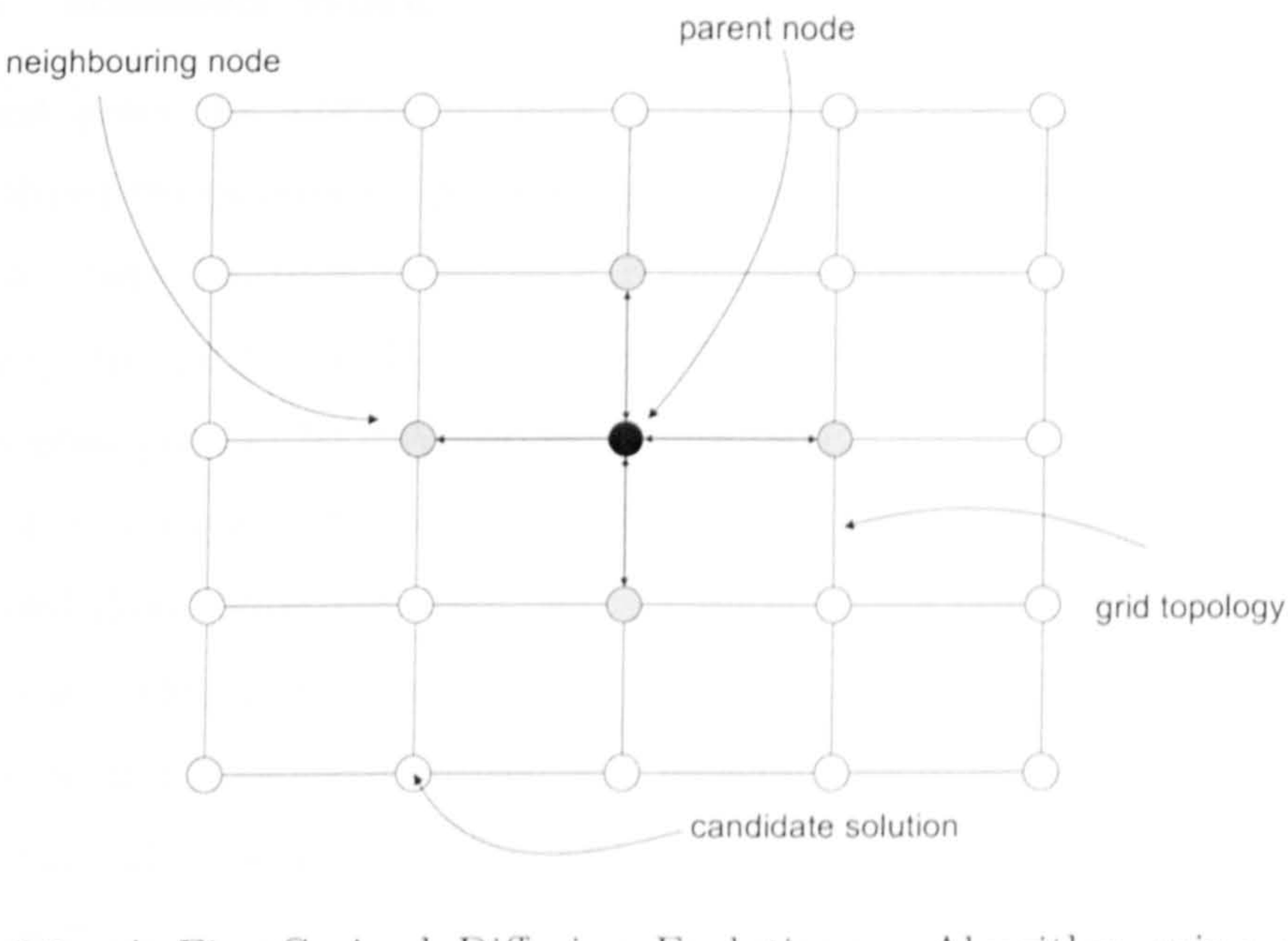


Figure 5.3: A Fine-Grained Diffusion Evolutionary Algorithm using a Grid Topology

The decision as to which of these forms of parallelism to implement must consider several factors, including ease of implementation and the potential per-

formance gains from parallelism. Single-population parallel EAs are often easiest to implement and use, since experience gained with sequential algorithms is directly applicable. In contrast, the implementation of parallel EAs with multiple communicating populations requires the consideration of extra design choices. For instance, the use of an island model EA requires the algorithm designer to choose the number of demes, the population topology, and the mutation rate, as well as choosing values for the standard evolutionary parameters. This increases the complexity of the parallel EA since each of these parameters influences the efficiency of the algorithm and the quality of the overall solution. However, as explained previously, a correct choice of parameters and topology for these multiple communicating population EAs can result in improved convergence of the algorithm for certain problems.

5.2.2 Related Work

In recent years the interest in using parallel evolutionary algorithms to solve single-objective optimisation problems has increased considerably (Alander 2003). However, there has been little research performed in applying parallel EAs to multi-objective problems. Hiroyasu *et al.* (2000) and Deb *et al.* (2003) have both implemented parallel MOEAs, although the MOEA implemented by Deb *et al.* (2003) is only suitable for problems with convex Pareto-optimal fronts (due to the guided domination approach used for dividing the population into demes), whilst Van Veldhuizen *et al.* (2003) present a thorough theoretical analysis and discussion of the issues surrounding development of parallel multi-objective evolutionary algorithms. However, none of these papers attempts to implement a parallel EA in a large-scale distributed environment such as a computational Grid.

Tan *et al.* (2003) and Fung *et al.* (2004) have both developed parallel evolutionary computing environments in Java to solve single-objective optimisation

problems. However, neither of these environments is well suited for use in a large-scale, multi-site computational Grid. The distributed evolutionary computing system proposed by Tan *et al.* (2003) is based on the island model of parallel EAs (see Section 5.2.1) and uses a small number of peers to host multiple communicating populations. Whilst in theory the island model should scale well to larger numbers of peers, Fernandez *et al.* (2003) have shown that this scalability is difficult to exploit in practice. Fung *et al.* (2004) propose a Java-based parallel platform for evolutionary computation using a Distributed Shared Memory (DSM) architecture (see Section 3.2.2). This architecture is unsuitable for use in a large-scale, multi-site computational Grid due to its tightly coupled nature.

Tanimura *et al.* (2002) have proposed a middleware system for enabling evolutionary optimisation in a Grid computing environment. This system requires the application designer to develop suitable evolutionary operators and implement them according to a common set of interfaces. Tanimura *et al.* (2002) use this middleware system to solve a single-objective optimisation problem by constructing a parallel simulated annealing algorithm. Abdalhaq *et al.* (2002) also use the concept of Grid computing to perform single-objective optimisation using evolutionary computation by developing a *Black Box Optimisation Framework (BBOF)* in C++ to optimise a computer simulation of a single-objective forest fire propagation problem. This optimisation process is run on a single compute cluster managed by the Condor resource management system (see Section 4.2.2).

Song *et al.* (2004) have implemented a single-objective genetic algorithm in a service oriented architecture to solve a 2D aerodynamic design optimisation problem. This approach is similar to that taken later in this Chapter; however, the distributed evaluation of candidate solutions in Song *et al.* (2004) is performed using a single compute cluster located at a single site, whereas the evaluation of candidate solutions described in this Chapter uses computational resources located at multiple geographically distributed sites. The evolutionary algorithm

implemented by Song *et al.* (2004) also focusses solely on single-objective optimisation.

The power of computational Grids is used by Luna *et al.* (2004) to perform a multi-objective distributed enumerative search. This search is used to generate the true Pareto optimal fronts for several benchmark test functions commonly used in the performance evaluation of multi-objective optimisation algorithms. Enumerative search is less computationally efficient than evolutionary optimisation methods, since enumerative search relies on evaluating every possible candidate solution in the search space. Luna *et al.* (2004) also briefly compare their enumerative search with results obtained using micro-GAs (Coello and Pulido 2001).

5.3 Comparison of the Island and Global Models of Parallel Evolutionary Computation

5.3.1 Experimental Set-Up

The performance of single population, globally parallel evolutionary algorithms and EAs using the island model of parallelism (see Section 5.2.1) is compared in this Section for a variety of single and multi-objective test functions taken from the literature. The baseline EA configuration used in the following experiments is shown in Table 5.1.

A key issue in the design of island model parallel EAs is choosing the extra parameters such as migration interval, migration rate, and the number and size of the subpopulations. For the island model EAs used in the following experiments general guidelines from the literature were used where possible. However, no general guidelines were found for the number and size of the subpopulations, so the total population was kept constant and the number of subpopulations varied between 1 (corresponding to the single population, globally parallel EA) and 10.

Total Population Size	100 individuals
Total Generations	250
Representation	Real-valued decision variables
Selection	Stochastic Universal Sampling
Variation	Bounded Intermediate Recombination Breeder Genetic Algorithm Mutation

Table 5.1: Baseline EA Configuration

The island EA with the number of subpopulations giving the best performance was then used for comparison.

Since evolutionary algorithms are stochastic processes, 100 runs were conducted with each algorithm so as to reduce the level of stochastic noise in the results and allow accurate performance comparisons between single population and island model EAs to be made. The performance of each algorithm run was measured using appropriate metrics for the test function under consideration: for single objective problems the distance of the best solution found by the algorithm from the true optimum value was used, whilst for multi-objective problems specific aspects of the quality of the approximation set (such as proximity to the true Pareto front and the diversity in the population) were measured using unary indicators from the literature[1].

A statistical comparison between the island model EA and the globally parallel algorithm was performed by computing the *t*-values[2] of the performance results produced by the algorithms. The significance of these results were then analysed using *randomisation testing*. The main advantage of randomisation testing is that it is a non-parametric test and therefore does not require any assumptions to be made about the data (Manly 1991). The basic premise is that, if the

[1]A thorough review of the use of performance indicators in evolutionary multi-objective optimisation is provided in Deb (2001), and some limitations of unary performance metrics pointed out in Zitzler *et al.* (2003).

[2]The *t*-value is the difference between the means of the datasets divided by the standard error.

null hypothesis is true (i.e. that any difference in performance has arisen by chance), then the observed result will appear as a typical value in many random resamplings of the data. The randomisation test proceedure is outlined below:

1. Compute the t-value of the two datasets. This is the observed result.
2. Randomly reshuffle the data and divide into two sets. Then recompute the t-value.
3. Repeat step 2 a large number of times to obtain the randomised distribution.
4. If the observed result appears a typical value in this randomised distribution, the accept the null hypothesis as true. Otherwise consider the alternative hypothesis (i.e. that one algorithm has outperformed the other). If the observed result appears in the top 5% of the randomised distribution it is said to be 'significant at the 5% level'.

In the following experiments, the results of this randomisation testing is shown graphically. Figure 5.4 illustrates a typical randomisation test. The randomised distribution is shown as a histogram and the observed result is shown as an asterisk on the x axis. An observed result to the left of the histogram indicates that set A outperforms set B, whilst an observed result to the right indicates the opposite is true (since the smaller the t-value the better the performance of set A). An observed result towards the middle of the histogram indicates that the null hypothesis is true. In the following experiments set A represents the globally parallel EA, and set B represents the island model EA.

5.3.2 Single-Objective Performance Comparison

Test Functions

The test functions used in the following experiments are a representative subset of those proposed for the Special Session on Real-Parameter Optimisation at

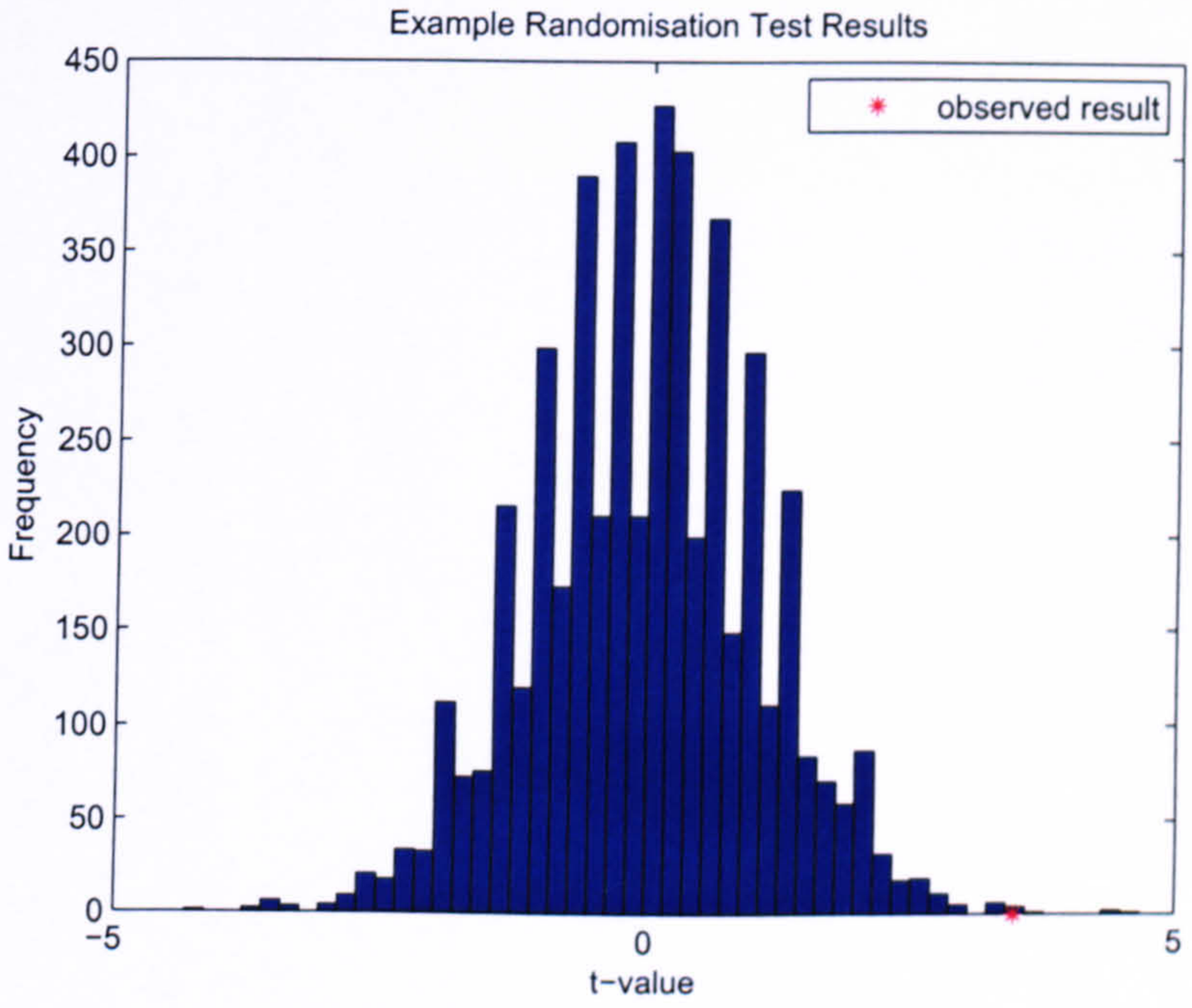


Figure 5.4: An Example Randomisation Test. Set B Outperforms Set A.

CEC2005 (IEE 2005). The test suite (Suganathan *et al.* 2005) consists of 25 unimodal and multimodal test functions designed to imitate some of the characteristics commonly found in real-world problems. Results from 13 of these test functions (shown in Table 5.2) are presented here.

Results

The single-objective island model EA from this Section used the migration strategy from Starkweather *et al.*'s (1991) GENITOR genetic algorithm. This uses an interconnected ring topology, where migration is performed every 5 generations. The migration rate was set to 10% of the population, since Cantú-Paz (1999) has shown that migration rates greater than this offer no significant improvement in convergence.

It can be seen from Figure 5.5 that the island model evolutionary algorithm exhibits better performance than the single-population, globally parallel EA on all

Test Function	Characteristics
Shifted Sphere Function	Unimodal, Shifted, Seperable, Scalable
Shifted Schwefel's Problem 1.2	Unimodal, Shifted, Non-seperable, Scalable
Shifted Rotated High Conditioned Elliptic Function	Unimodal, Shifted, Rotated, Non-seperable, Scalable
Shifted Schwefel's Problem 1.2 with Noise in Fitness	Unimodal, Shifted, Non-seperable, Scalable, Noise in Fitness Function
Schwefel's Problem 2.6	Unimodal, Non-seperable, Scalable
Shifted Rosenbrock's Function	Multi-modal, Shifted, Non-seperable, Scalable, Narrow Valley from Local Optimum to Global Optimum
Shifted Rotated Ackley's Function	Multi-modal, Rotated, Shifted, Non-seperable, Scalable
Shifted Rastrigin's Function	Multi-modal, Shifted, Seperable, Scalable, Huge Number of Local Optima
Shifted Rotated Rastrigin's Function	Multi-modal, Shifted, Rotated, Non-seperable, Scalable, Huge Number of Local Optima
Shifted Rotated Weierstrass Function	Multi-modal, Shifted, Rotated, Non-seperable, Scalable
Schwefel's Problem 2.13	Multi-modal, Shifted, Non-seperable, Scalable
Expanded Extended Griewank's plus Rosenbrock's Function	Multi-modal, Shifted, Non-seperable, Scalable
Shifted Rotated Expanded Scaffer's F6	Multi-modal, Shifted, Non-seperable, Scalable

Table 5.2: Single Objective Test Functions and their Characteristics

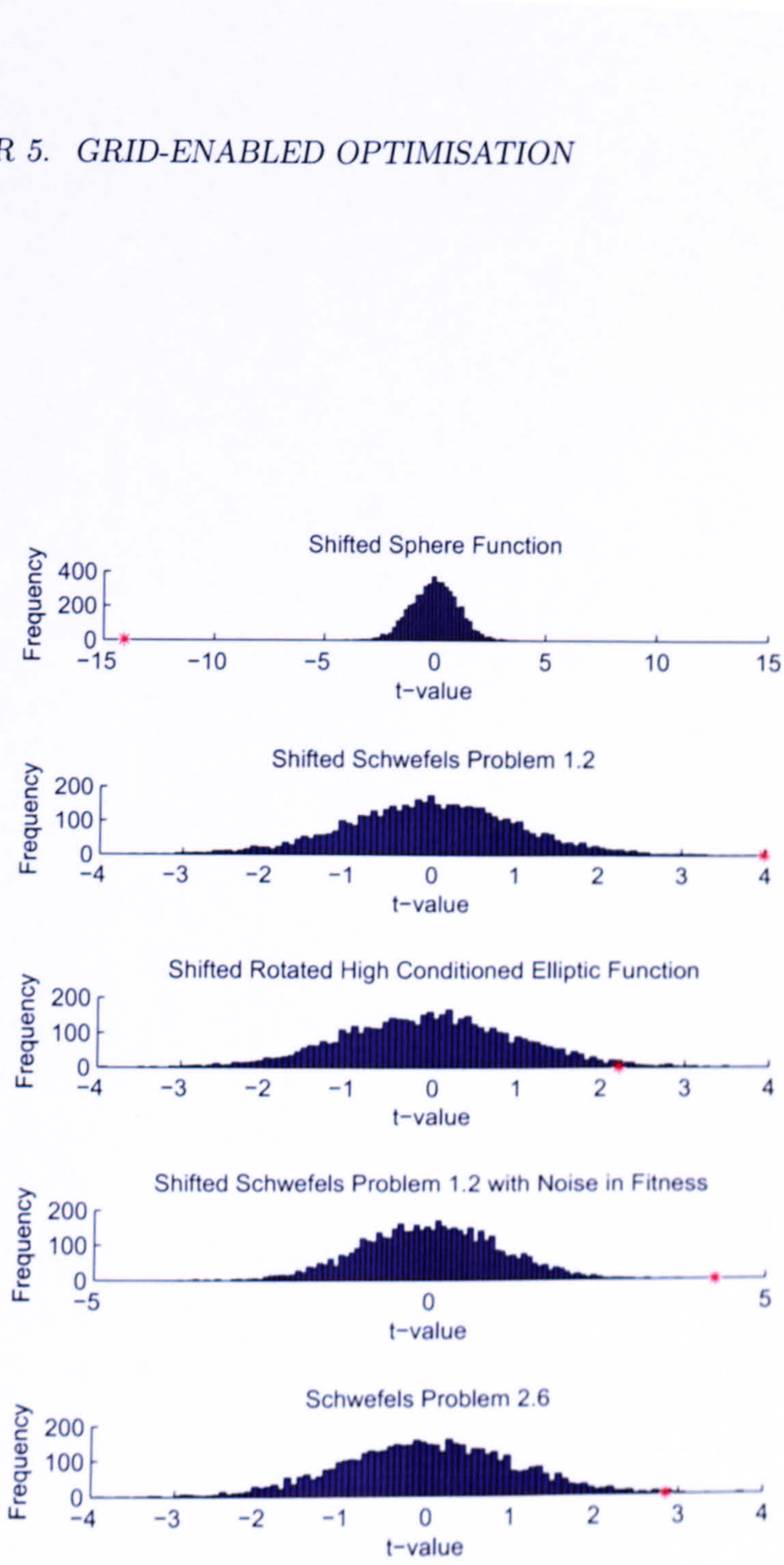


Figure 5.5: Randomisation Data for the Unimodal Test Functions

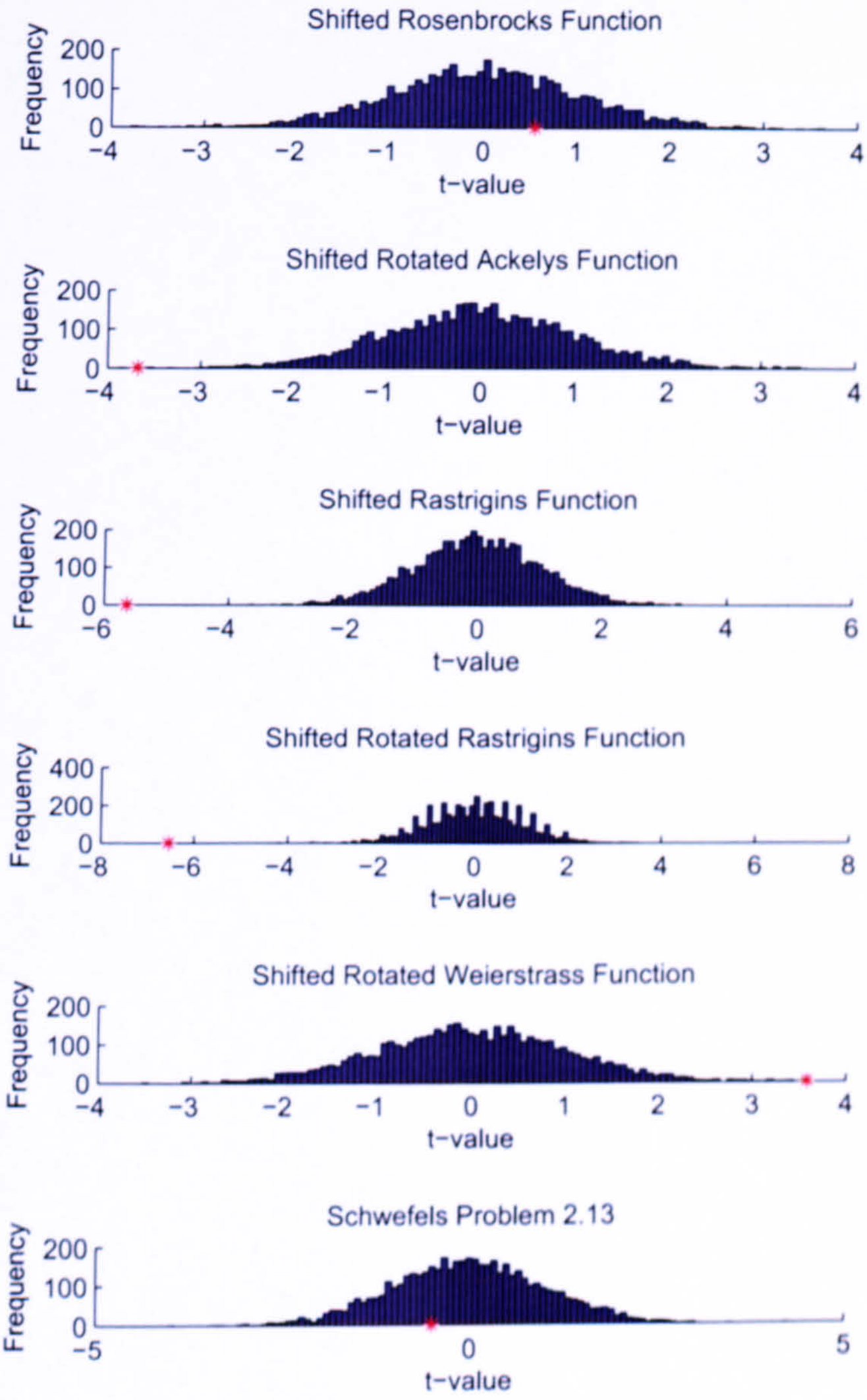


Figure 5.6: Randomisation Data for the Multimodal Test Functions

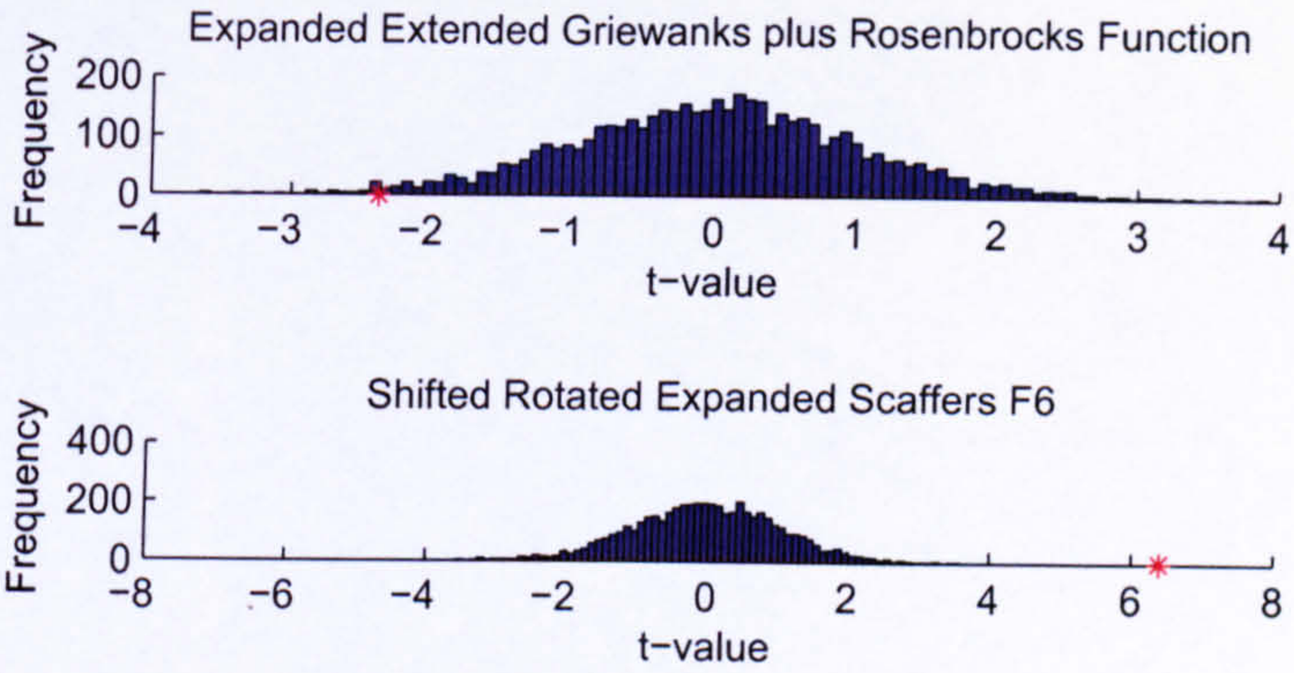


Figure 5.7: Randomisation Data for the Expanded Multimodal Test Functions

but one of the unimodal test functions. However, Figure 5.6 shows that the single population algorithm performs better on most of the multimodal test functions. It is only on the Shifted Rotated Weierstrass Function that the island model EA outperforms the globally parallel algorithm. Figure 5.7 shows that the globally parallel algorithm performs better than the island model EA for the Expanded Griewank's plus Rosenbrock's Function, whilst the opposite is true for the Shifted Rotated Expanded Scaffer's F6 Function.

Conclusions

These results suggest that, in multimodal fitness landscapes, the island model EA often converges to local optima. This may be because, as Wright (1932) suggested in the field of population genetics, smaller subpopulations are more susceptible to random genetic drift (something that the selection process opposes in large panmictic populations). Whilst in unimodal fitness landscapes it is possible that this random drift helps the island model EA explore the search space, in multimodal problems with many local optima it may result in the subpopulations drifting away from previously discovered good solutions and towards those local optima. Figures 5.8 and 5.9 show the effects of this random drift on a two variable

version of Rastrigin's function[3]. As Figure 5.8 shows, by the tenth generation the globally parallel EA is starting to converge, whilst the island model EA is still exploring the search space. At the twentieth generation (Figure 5.9), the single population EA has converged to the globally optimal solution, whilst most of subpopulations in the island model EA are still drifting towards local optima.

5.3.3 Multi-Objective Performance Comparison

Test Functions

The benchmark set of multi-objective test problems developed by Zitzler *et al.* (2000) is used in the following study. These functions have well defined Pareto optimal fronts, and contain many features commonly found in real-world multi-objective problems (see Table 5.3).

Test Function	Characteristics
ZDT1	Convex Pareto-optimal front
ZDT2	Non-convex Pareto-optimal front
ZDT3	Discontinuous Pareto-optimal front
ZDT4	Many local Pareto-optimal fronts
ZDT5	Discrete variable representation with a deceptive Pareto-optimal front
ZDT6	Non-convex Pareto-optimal front with a non-uniform distribution of Pareto-optimal solutions

Table 5.3: Multi Objective Test Functions and their Characteristics

Results

The island model MOEA used in this section was based on the Divided Range Multi-Objective Genetic Algorithm (DR-MOGA) proposed by Hiroyasu *et al.*

[3]The global optimum of this solution is at [0, 0], and 8 other local optima exist at [1, -1], [-1, 0], [1, 1], [0, -1], [0, 1], [-1, -1], [-1, 0], and [-1, 1].

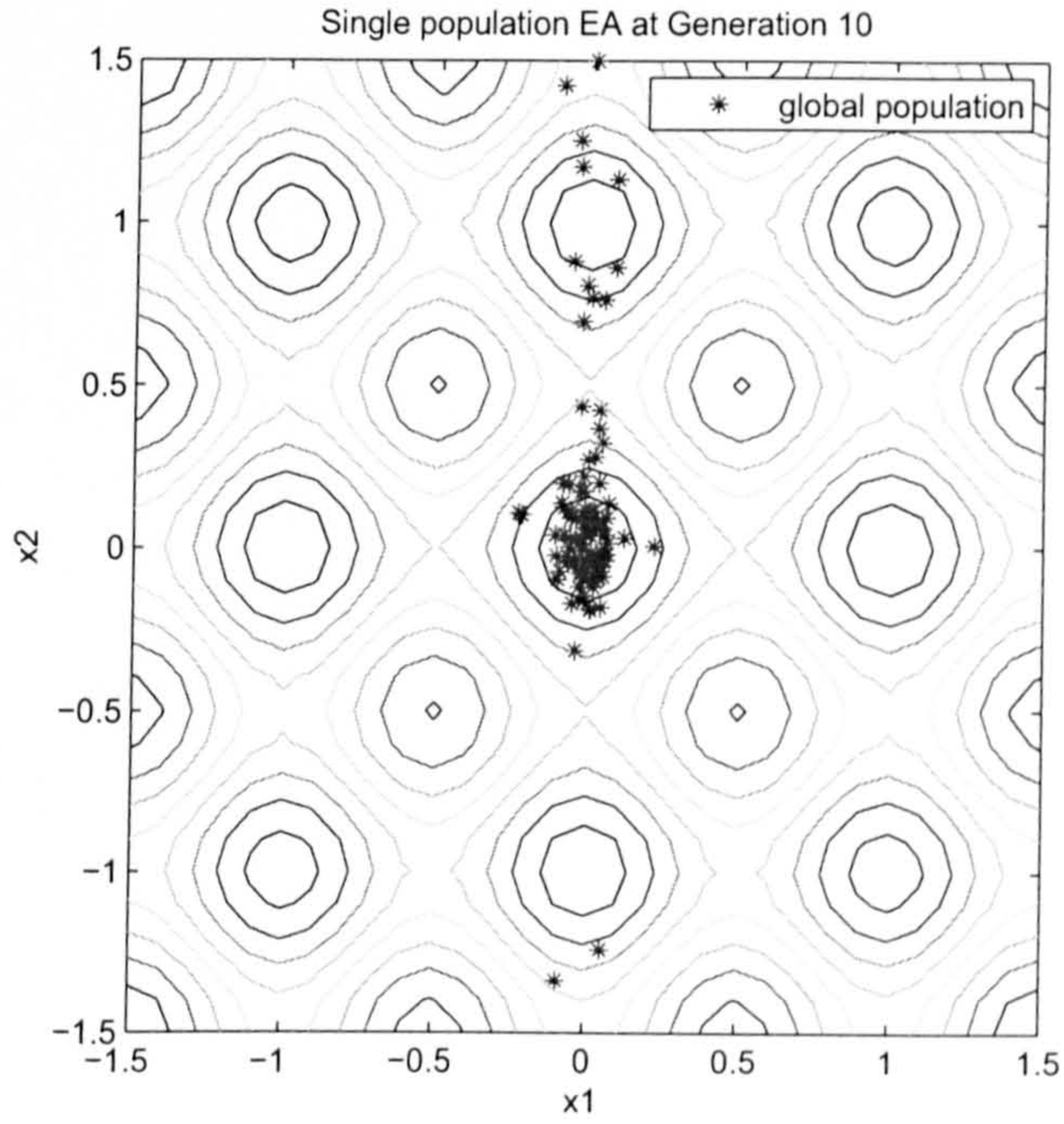
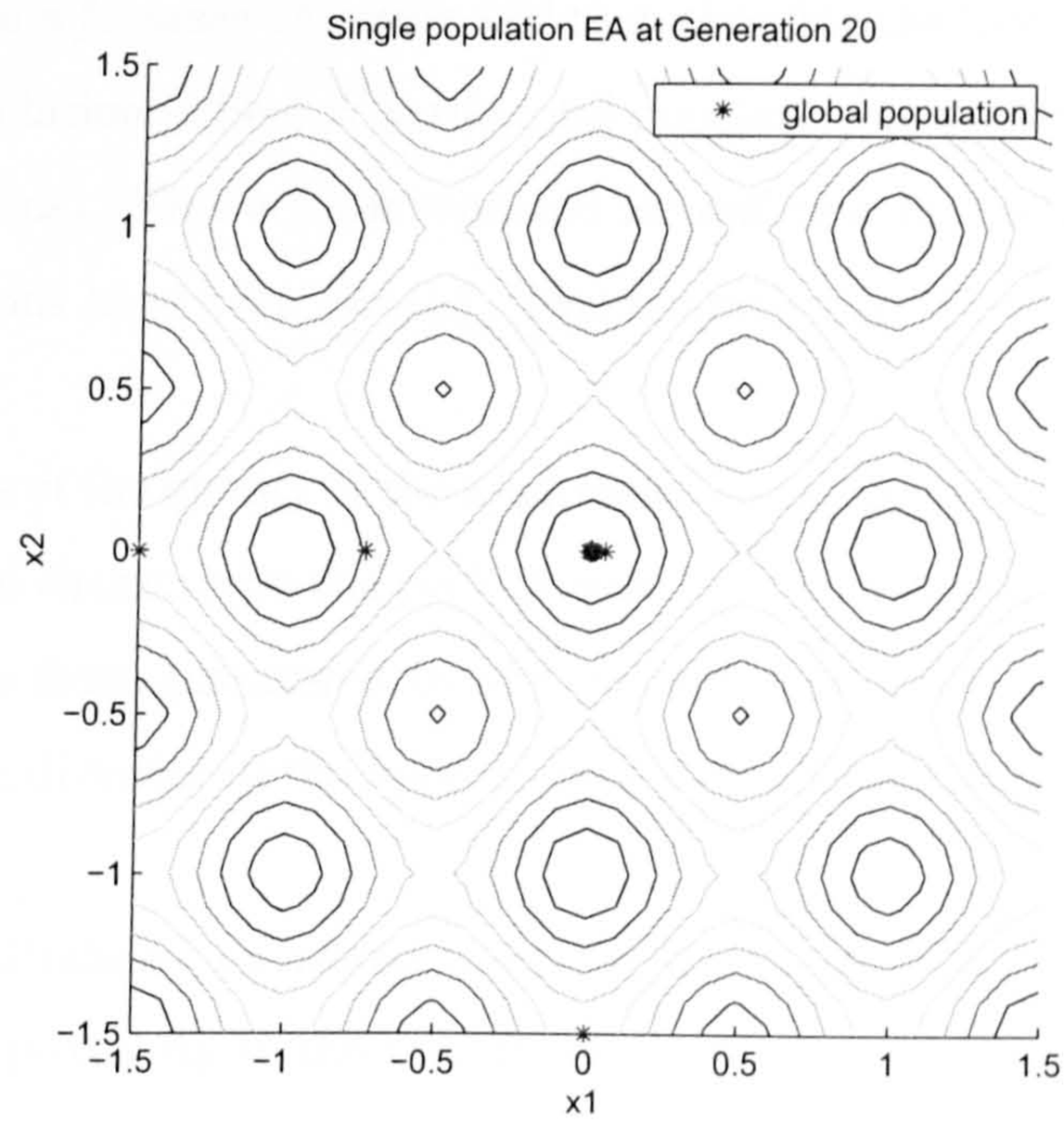


Figure 5.8: A Comparison of the Search Landscapes of the Panmictic and Island Model EAs after 10 Generations



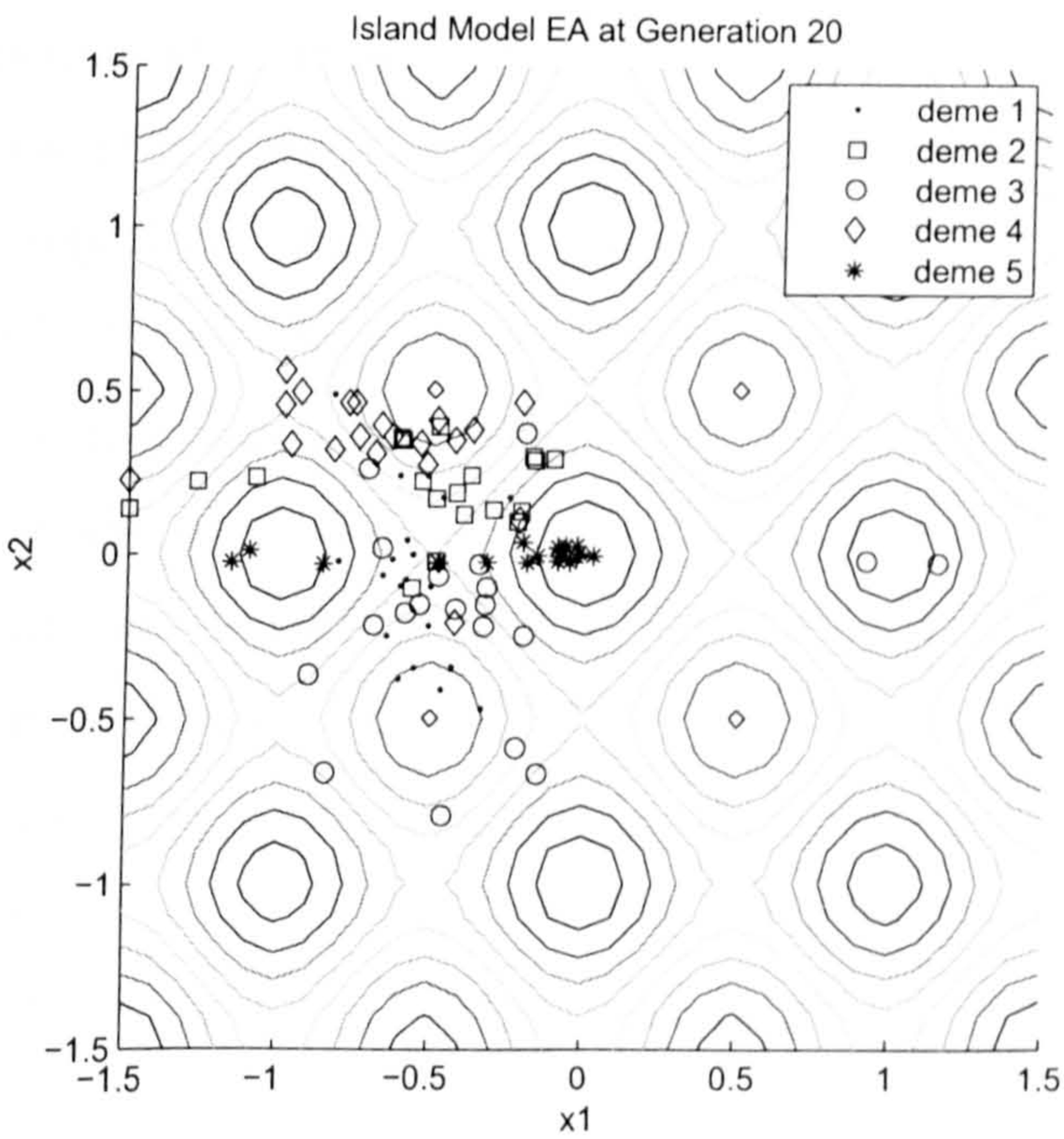


Figure 5.9: A Comparison of the Search Landscapes of the Panmictic and Island Model EAs after 20 Generations

(2000). In this algorithm migration is performed by sorting the approximation set according to a *focussed objective*, and then choosing the best N/m solutions to form a subpopulation (where N is the total population size, and m is the number of subpopulations). This is then repeated for different focussed objectives until m subpopulations have been created. This migration process is performed every 5 generations.

Two aspects of the quality of the approximation set produced by the optimiser are used here to characterise the performance of the algorithm: the proximity to the true Pareto front (measured by the *generational distance* (Van Veldhuizen 1999b)) and the diversity of the approximation set (measured by the *spread* (Deb *et al.* 2000)).

As Figure 5.10 shows, the island model MOEA is able to find an approximation set with better proximity to the true Pareto-optimal front for four of the multi-objective test functions investigated in this Section. The test functions that the single-population, globally parallel MOEA performed better on were ZDT-1, where the Pareto-optimal set is convex, and ZDT-6, where the Pareto-optimal set is not uniformly distributed due to the search space for the first objective being multimodal. It can also be seen from Figure 5.11 that the island model MOEA obtains a more diverse approximation set than the globally parallel MOEA on two of the problems, whilst the globally parallel MOEA obtains a more diverse approximation set on ZDT-1. Interestingly, on all but one of the test functions where the island model MOEA obtained better proximity to the true Pareto-optimal front less diverse approximation sets were found, confirming previous findings that the goals of obtaining both proximity and diversity in the approximation set are in conflict (Bosman and Thierens 2003).

Conclusions

These results tend to confirm the findings from the single-objective investigation in Section 5.3.2 that the globally parallel algorithm exhibits better performance

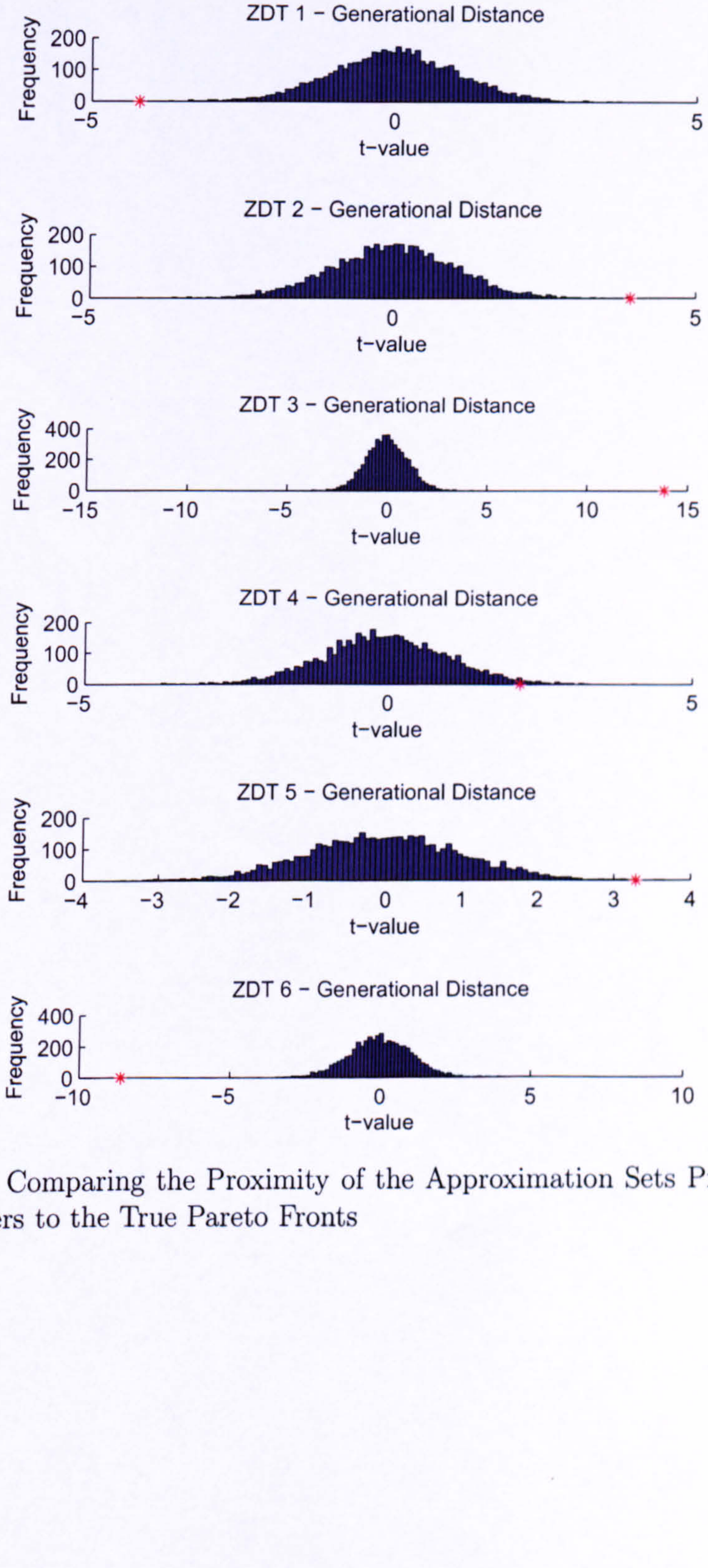


Figure 5.10: Comparing the Proximity of the Approximation Sets Produced by the Optimisers to the True Pareto Fronts

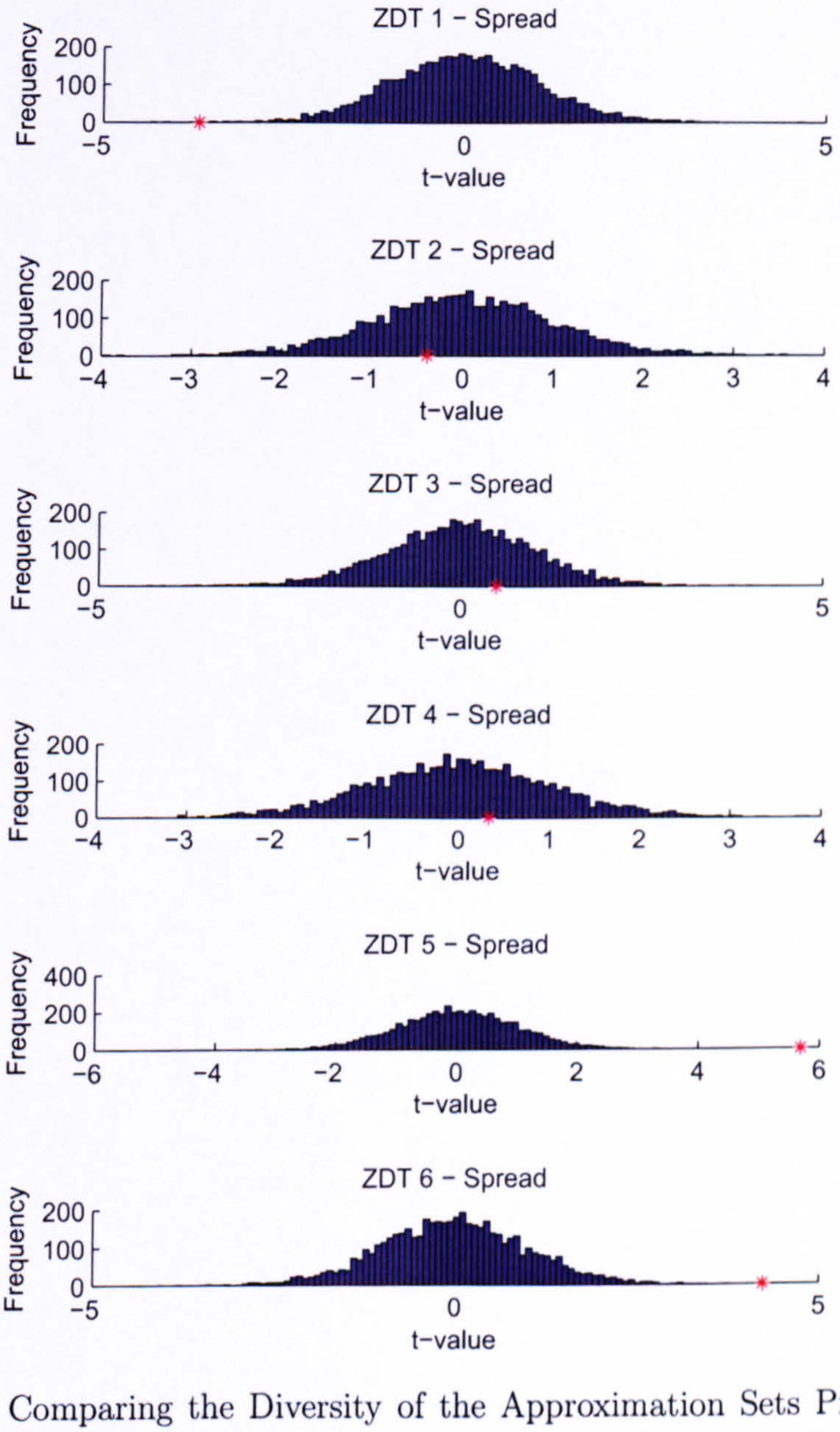


Figure 5.11: Comparing the Diversity of the Approximation Sets Produced by the Optimisers

in multimodal search spaces than the island model algorithm, since the globally parallel MOEA performed better than the island model MOEA on ZDT-6. However, only one of the multi-objective test problems examined in this Section required searching a multimodal fitness landscape and so it is not possible to make a general assertion that this result will always hold for multi-objective problems.

These results show that the island model MOEA using the divided range migration strategy obtains better proximity than the globally parallel MOEA on several of the test problems studied. This may be because the reordering of the solutions in the migration step prevents the formation of *lethals*[4] in a similar way to using a mating restriction (Deb and Goldberg 1989), since the individuals in each subpopulation inhabit a similar area of the search space.

5.4 Introducing a Grid Based Framework for Evolutionary Optimisation

5.4.1 Parallelisation of the Evolutionary Algorithm

In Section 5.2.1 two types of possible parallelisation strategies for evolutionary algorithms were found: multiple communicating populations and single-population master-slave implementations. Section 5.3 presented a comparison of these two techniques for a variety of single and multi-objective problems taken from the literature, and it was found that different types of parallelisation suit different types of problems. Section 5.3 shows that the island model EA generally performed well on unimodal problems, whilst the single-population globally parallel algorithm performed better in multimodal search spaces.

One major drawback with the island model EA was the difficulty in setting the additional parameters for the algorithm. Although guidelines exist in the

[4]Lethals are individuals of low fitness that can be created by the recombination of parents from dissimilar areas of the search space.

literature for setting the migration rate and migration interval, no general guidelines were found for the optimal number of subpopulations to use. In fact, the results from Section 5.3 suggest that the optimal number of subpopulations varies from problem to problem. For example, the optimal number of demes for the shifted rotated Weierstrass function in Section 5.3.2 was 4, whilst for Schwefel's problem 1.2 it was 2.

In the multi-objective case the optimal number of subpopulations often also varied according to which aspects of approximation set quality were of interest. For example, on the ZDT-4 problem, the number of demes giving the best proximity to the true Pareto-optimal front was 5, whilst the best diversity in the approximation set resulted from 2 subpopulations.

The experiments in Section 5.3 have shown that the number of demes is an important factor in the quality of the solutions produced by a parallel evolutionary algorithm. However, in a Grid computing environment this number of demes and population structure may be determined by the configuration of the Grid resources and may therefore not be the optimal number for the problem under consideration. For this reason it was decided to use the single-population master-slave implementation, since it does not require the choice of these extra parameters. The single-population model also allows experience from implementing sequential EAs to be easily transferred.

5.4.2 Optimisation in a Service-Oriented Architecture

The Grid-based framework for evolutionary optimisation introduced in this Chapter is implemented in a Service-Oriented Architecture (SOA) using Grid Services based on the Globus Toolkit (see Section 3.3) to provide access to the resources in the Grid. This framework is written using the Java programming language so as to provide portable code that allows components of the framework to be easily run across various heterogeneous platforms.

A service-oriented architecture is essentially a collection of services that communicate with each other in order to perform a complex task. SOA is an approach to building loosely-coupled, distributed systems that combine services to provide functionality to an application. IBM sees SOA as the key to meeting interoperability and flexibility requirements for its vision of an on demand business (Colan 2004).

The service-oriented architecture approach to enabling Grid computing is well suited to the kind of master-worker parallelism implemented in the evolutionary optimisation framework described in this Chapter. In a service-oriented view of Grid computing the client application acts as the master node, whilst the Grid service performs the role of the worker nodes. The framework presented in this Chapter (see Figure 5.12) uses two different types of service. One service type exposes the evolutionary algorithm operators (such as ranking, selection and variation) to the client application, and the other provides the ability to run objective function evaluations on the available Grid resources.

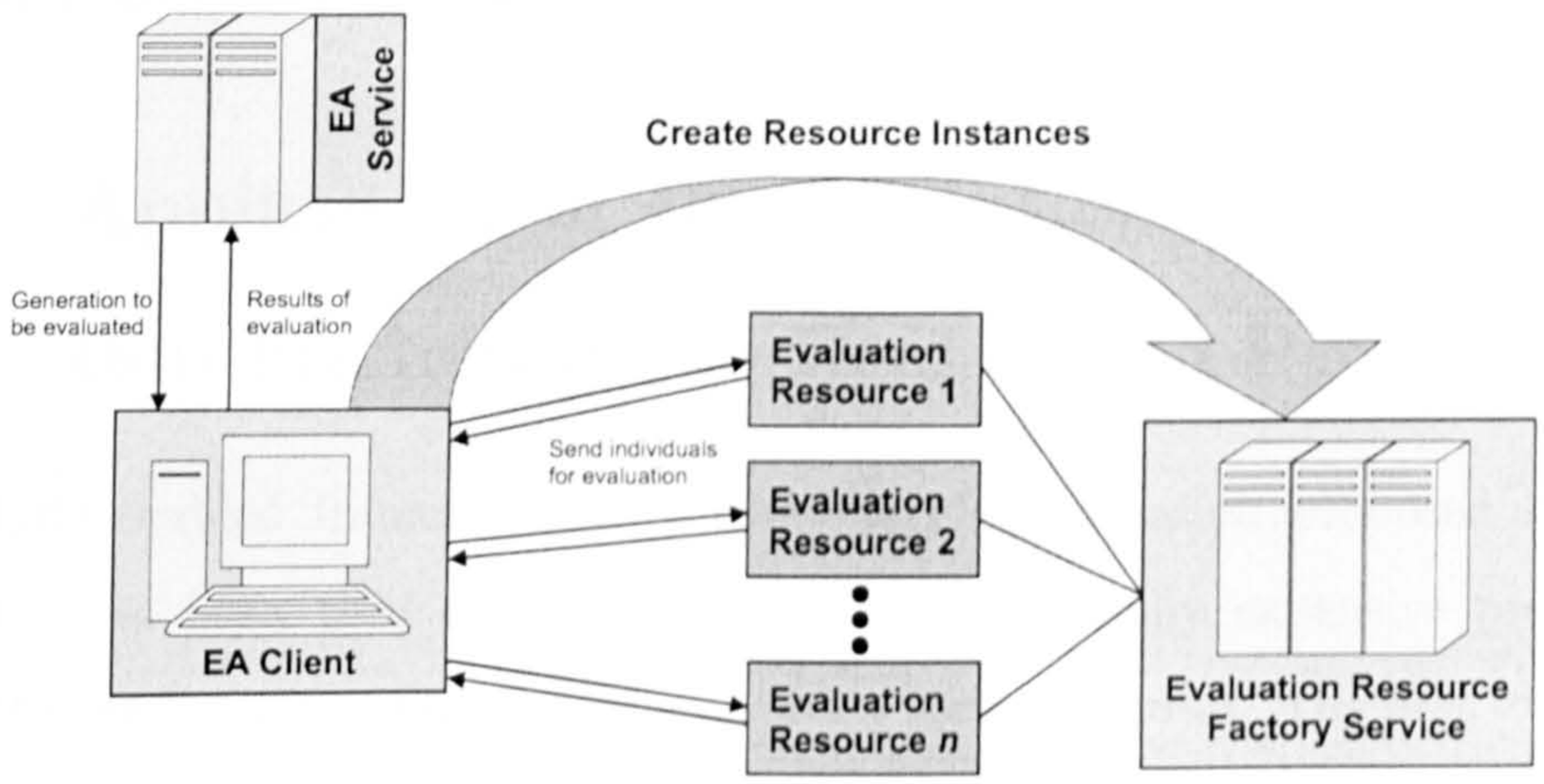


Figure 5.12: The Implementation of the Optimisation Framework

The Evaluation Resource Factory Service shown in Figure 5.12 implements the Grid-based meta-scheduling architecture developed in Chapter 4 to provide access to the available Grid resources. Job scheduling is performed using the novel

hybrid approach proposed in Section 4.6 to optimally allocate each generation of candidate solutions for evaluation amongst the Grid resources using response time information gathered from the previous generation. Using this hybrid adaptive approach allows the Evaluation Resource Factory Service to overcome any bottlenecks in the scheduling of jobs by the Grid resources.

This SOA approach also provides a great deal of flexibility both in how the optimisation framework is used and in its maintenance. The provision of the components of the framework as services means that it is simple to add new functionality to the system, and to improve upon existing functionality, by adding new services. In the context of the optimisation framework, this functionality could be anything from the implementation of additional evolutionary operators to the distribution and management of objective function evaluations. The SOA approach taken here also allows the easy configuration of this framework to perform either single or multi-objective optimisation by providing methods for both Pareto-based (for multi-objective optimisation) and fitness-based (for single-objective optimisation) ranking of candidate solutions.

5.5 Application of the Evolutionary Optimisation Framework to Real World Problems

The Grid-enabled framework for evolutionary optimisation proposed in this Chapter has been used to optimise two computationally expensive problems currently under investigation in the author's research group. The first example presented in this Section is a single-objective aero-engine maintenance problem that, as explained in Section 5.5.1, has the potential to significantly impact on Rolls-Royce's business strategy, whilst the second is a multi-objective controller design problem that shows the versatility of the framework. These real world examples serve to highlight the usefulness of the proposed framework in tackling

large-scale complex problems.

5.5.1 An Aero-Engine Maintenance Scheduling Example

A fundamental shift in emphasis within the aero-engine manufacturing industry is leading to the adoption of power-by-the-hour contracts, where airlines make regular fixed payments to the engine manufacturers based on the hours flown by an engine and, in return, the manufacturers of the engine retain the responsibility for servicing and maintenance. As a result of this, the accurate prediction of support costs over the life-cycle of an engine is of the up-most importance. However, aero-engines operate in a highly complex and unpredictable environment, and as such it is impossible to produce a deterministic model for these support costs. Instead, stochastic simulations can be performed to provide cost estimates. It is important for the engine manufacturers to devise maintenance scheduling strategies to minimise support costs and thus enable more competitive pricing of these contracts.

Life-Cycle Simulation of Aero-Engines

The Modular Engine Arisings, Repair and Overhaul Simulation (MEAROS) package was developed to enable Rolls-Royce and the Ministry of Defence to evaluate the operation, maintenance and supply of aircraft engines (Rolls-Royce 2002). Although designed for the aero-engine manufacturing industry, the simulation can equally be applied to ships, land vehicles and power generation (Argyle 2006).

The data collected during a simulation run falls into three main categories:

- **Operations Reports** which describe the nature of the operations undergone by the engines in the simulation (such as number of hours flown).
- **Arisings Reports** which detail events that cause an engine to be taken out of service.

- **Maintenance Reports** which detail what maintenance actions were taken (such as the reconditioning or scrapping of engine modules) and the times that these occurred.

The modelling capability of the MEAROS software is extensive, with the software able to model the operation of fleets of engines with an arbitrary number of modules (Rolls-Royce 2002). Theoretically there is no limit to the size of fleets that can be modelled by the software, however in practice this is limited by the computational effort needed to model large numbers of engines.

Results produced by the simulation contain a lot of stochastic noise due to the probabilistic models used to simulate component failures. As such, the simulation has to be run multiple times and averaged to reduce the effect of this noise. Figure 5.13 shows that the standard deviation of the aggregate maintenance cost reduces with the number of runs of the model. It can also be seen from Figure 5.13 that the improvement in the standard deviation tails off significantly after 100 passes. In practice this means that the benefit from running more than 100 passes of the model is outweighed by the additional computational cost.

Originally MEAROS was used for predicting the number of spares needed to maintain a set level of operational availability. However, many of the parameters in the model are customisable (such as the failure distributions of engine modules, the stock levels, and the maintenance scheduling strategies used) and can therefore be optimised with respect to some objective (for instance the support costs or the operational availability).

In this study optimisation was performed on the maintenance scheduling strategy with minimising the total cost of maintenance as the objective. The maintenance scheduling strategy was chosen because it is one of the few parameters affecting operational availability and support costs that is easily modifiable once an engine has gone into service. Maintenance in the simulation is performed after an arising occurs. The main causes of arisings are either the

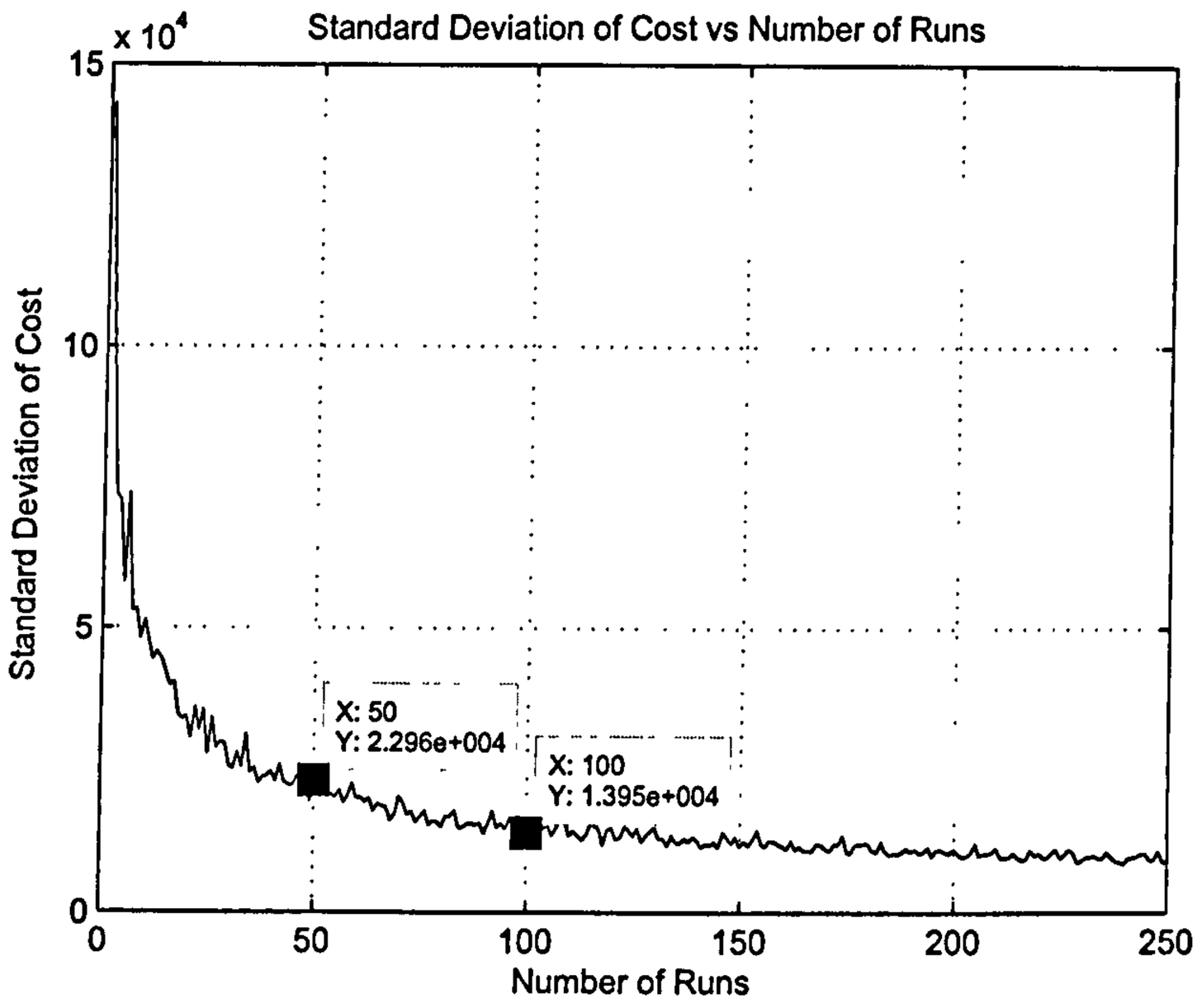


Figure 5.13: Standard Deviation of Aggregate Maintenance Cost Against the Number of Runs of the Model

expiry of a *hard-life*[5] or an in-service failure such as foreign object damage (Rolls-Royce 2002).

Once an arising occurs, the engine must be removed from the aircraft wing and the module that caused the arising must be reconditioned or replaced. However, as the removal of the engine from the wing is one of the most expensive parts of a typical maintenance shop visit, this provides the ground crew with the chance to perform opportunistic maintenance on the other modules in the engine. If one of the other modules in the engine has exceeded its *soft-life*[6] then it should also be reconditioned or replaced whilst the engine is removed from the wing. Crocker and Kumar (2000) have shown that, for relatively small engine module costs, there is likely to be an optimal set of soft-life values which minimise the maintenance cost of an engine. Soft-lives that are too low result in engine modules being reconditioned or replaced during every maintenance shop visit, whilst soft-lives that are too high result in cheaper but more frequent shop visits.

Table 5.4 shows a simple cost model developed in conjunction with Rolls-Royce for a hypothetical five module aero-engine (Argyle and Tubby 2002). The costs given represent the costs of removal and reconditioning the modules in the engine, whilst the scale and slope parameters represent the characteristic life of a module and the failure distribution of a module respectively.

	Cost	Scale	Slope
Engine	3000	N/A	N/A
Module 1	200	1000	1
Module 2	1000	800	2.5
Module 3	900	700	3
Module 4	800	2000	2
Module 5	1200	1500	1.5

Table 5.4: Cost Model and Weibull Distribution Parameters

[5]Hard-lives are usually assigned to safety critical components and represent the age at which that component must be replaced.

[6]Soft-lives represent the age whereby a component should be replaced at the next opportunity.

Development of Optimal Maintenance Scheduling Strategies

The framework proposed in this Chapter was used to optimise aero-engine maintenance scheduling strategies across a fleet of aircraft. As explained above, the maintenance scheduling strategy[7] was chosen for optimisation as it is one of the few parameters that is both easily modifiable once an engine has gone into service and has a significant impact on support costs and operational availability. It can also be quickly implemented across an entire engine range and is inexpensive to vary when compared to post-production design changes (Argyle and Tubby 2002).

The evaluation of candidate solutions in the optimisation process was performed using the MEAROS engine life-cycle model in conjunction with the simple cost model shown in Table 5.4. MEAROS was used to simulate the operation of a fleet of 25 engines over a 10 year period, and the results were averaged over 100 passes of the model so as to reduce the stochastic noise from the simulation (see Figure 5.13). Evaluation of a single candidate solution using this configuration took in the order of 1.5 seconds on a Intel Pentium 4 based PC running at 3.0GHz.

Evolutionary Algorithm Implementation

The Grid-enabled optimisation framework presented in this Chapter was used in a genetic algorithm architecture with the decision variables represented as real values. Fogel and Ghoziel (1997) have shown that there is no intrinsic advantage in choosing one bijective representation over another. Consequently modern EA practice emphasises choosing a representation that is appropriate for the problem under consideration. As the decision variables in the problem considered in this Section are continuous it is intuitive to use a real-valued

[7]The maintenance scheduling strategy optimised here is determined by the set of soft-lives for the modules in the engine.

representation (Michalewicz and Fogel 2000). Selection in the optimisation process was performed using Stochastic Universal Sampling (Baker 1987) which guarantees sampling with zero bias and minimum spread, whilst variation was introduced into the population using the extended intermediate recombination and BGA mutation operators (Mühlenbein and Schlierkamp-Voosen 1993). The EA also used a population size of 50 individuals.

Optimisation Results

The EA was run multiple times for the cost model provided in Table 5.4. However, as similar results were obtained from each execution of the algorithm, the following results are from a single representative run only. It can be seen from Figure 5.14 that the mean value of the population (shown as a solid line in the Figure) shows convergence after 15-20 generations. Figure 5.14 also shows that the diversity in the population[8] is significantly reduced as the search progresses.

Figure 5.15 shows the distributions of the engine module soft-lives from 30 generations after the optimiser has converged, with the quartile values marked on the plots by vertical lines. It can be seen from Figure 5.15 that both the soft-lives found by the optimiser and the interquartile ranges of their distributions are relatively high for modules 1 and 5. This is because the failure distributions for these modules indicate that they fail randomly[9] and nearly randomly, respectively. Assigning soft-lives to these components will therefore just increase the overall maintenance cost. Modules 2, 3, and 4 are all assigned lower soft-lives by the optimiser, indicating that preventative maintenance of these components can reduce the overall cost. Modules 2 and 3 should be replaced often, since they have relatively short characteristic lives, whilst module 4 has a much longer characteristic life and therefore should not be replaced as much. Analysis of the model output indicates that module 4 rarely causes arisings.

---

[8]Each individual in the population is shown in Figure 5.14 as a single point.

[9]A failure distribution with a slope of one indicates a component will fail randomly.

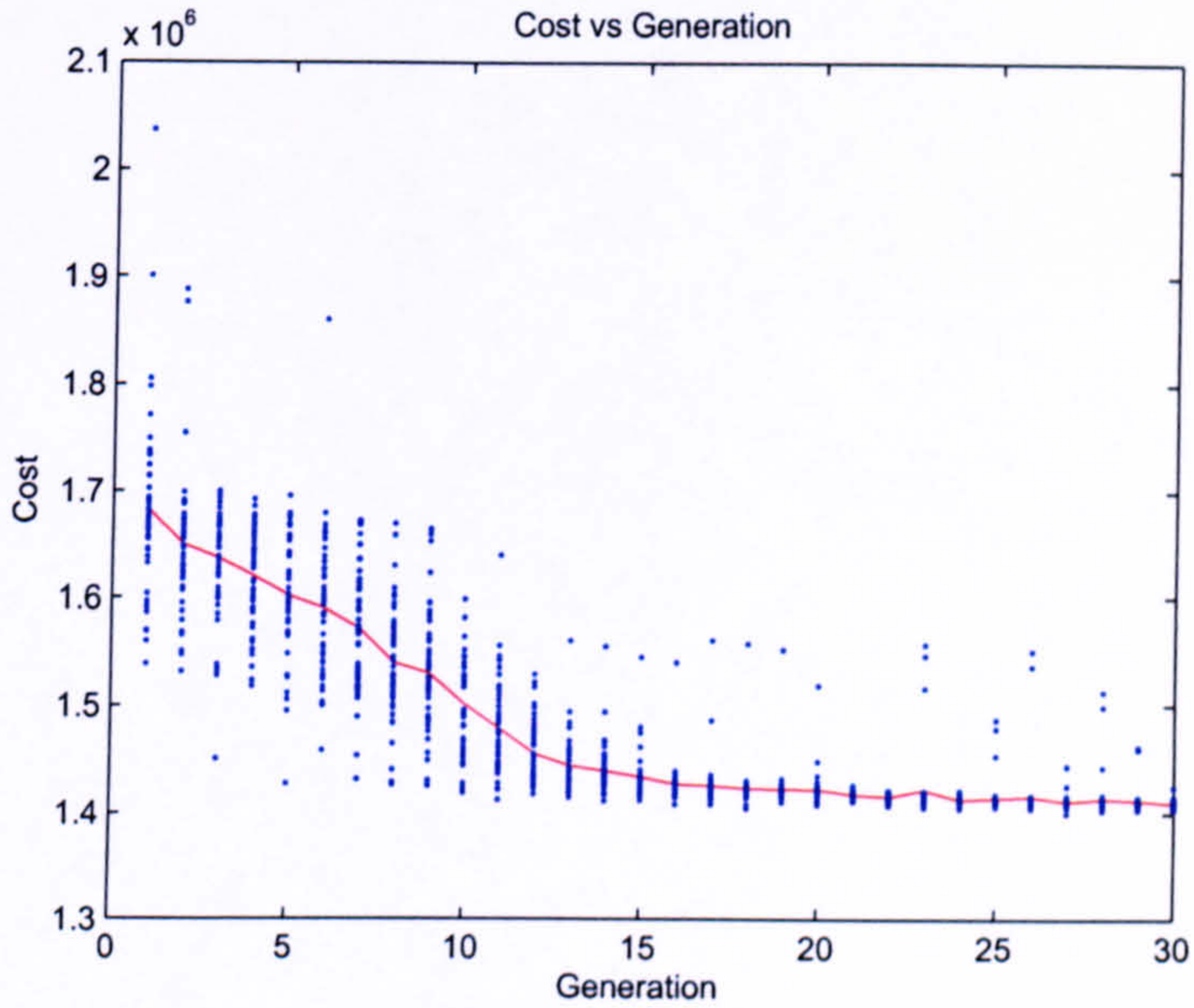


Figure 5.14: Cost of the Maintenance Scheduling Strategy Against the Number of Generations

Table 5.5 shows a representative set of total execution times from the optimisation of maintenance scheduling strategies for the aero-engine problem described in this Section, running for 30 and 50 generations respectively. These times are averaged over 5 runs for both single workstation results and the results obtained using the resources of the White Rose Grid (see Section 4.3). As Table 5.5 shows, the use of the Grid-based optimisation framework proposed in this Chapter has considerably reduced the time taken to optimise this aero-engine maintenance strategy problem.

Single Workstation		Computational Grid	
30 generations	50 generations	30 generations	50 generations
1981 seconds	3372 seconds	826 seconds	1354 seconds

Table 5.5: Execution Times for the Optimisation of Maintenance Scheduling Strategies

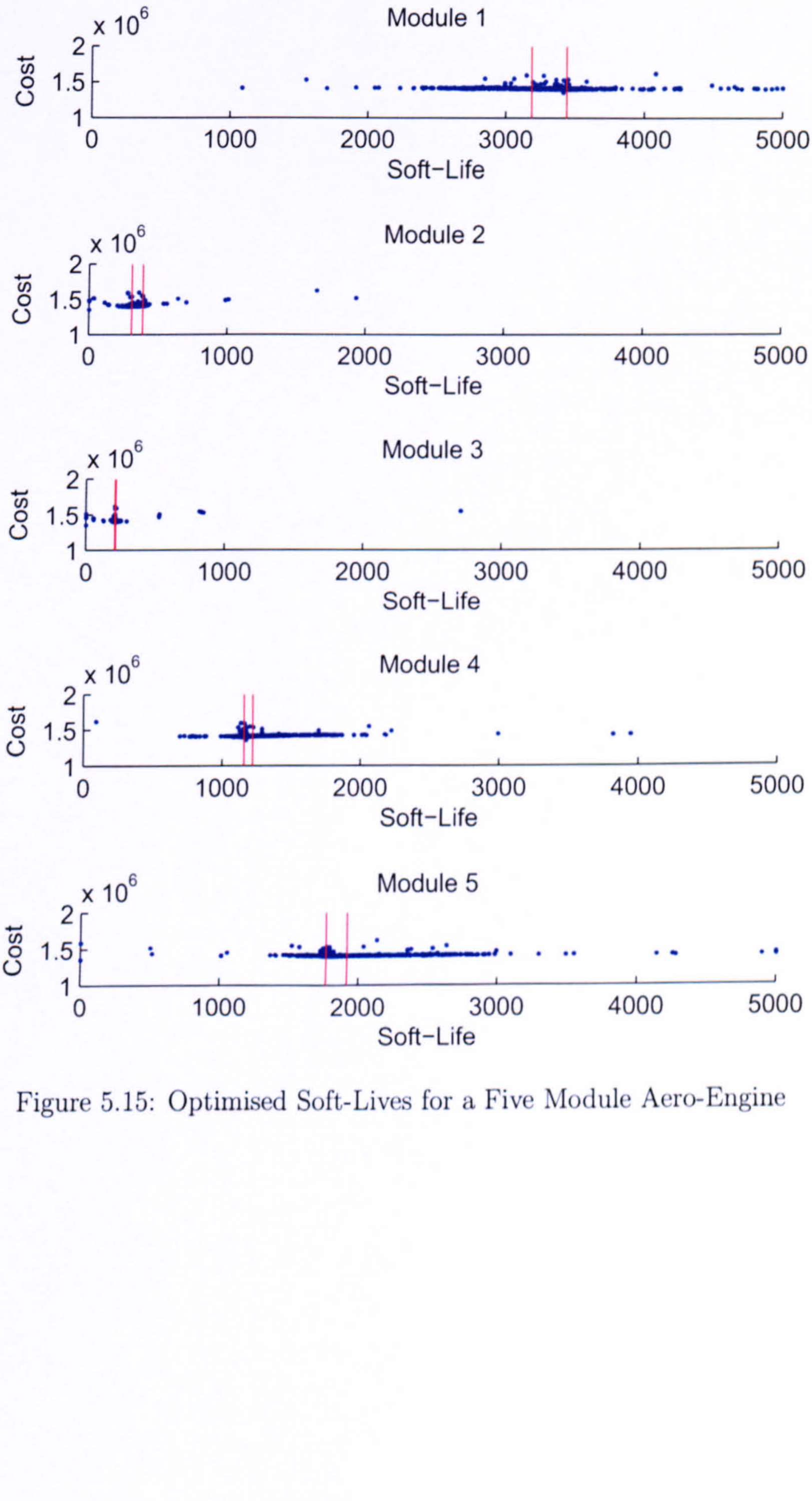


Figure 5.15: Optimised Soft-Lives for a Five Module Aero-Engine

Discussion

The aero-engine model used for the maintenance scheduling strategy optimisation presented here represents a simplified version of a real-world system. This is primarily due to computational constraints encountered in previous work (Argyle 2006). However, the speed up obtained using the Grid-enabled optimisation framework proposed in this Chapter will enable the optimisation of larger scale models, such as those applied to bigger fleets of aircraft or those with higher fidelity models of engines (i.e. comprising of a larger number of modules). This speed-up also enables a decision maker to run the optimisation process multiple times with alternative cost models to get a clearer understanding of the problem space.

5.5.2 An Aircraft Control Systems Design Example

Modern aircraft consist of many complex subsystems, all of which require robust and reliable control. These subsystems are often multi-variable, consisting of multiple inputs and multiple outputs, and frequently the desired responses of a subsystem (such as overshoot and rise time) are in conflict. Coupling evolutionary multi-objective optimisation techniques with conventional controller design methods such as H_∞ or LQG control can provide the engineer with a powerful tool for addressing such problems.

H_∞ Control of Aircraft Flight Dynamics

The control system for the flight dynamics of an aircraft must provide robust and responsive multi-variable control of the ailerons and the rudder, as well as guaranteeing stability in the presence of modelling uncertainty. H_∞ control theory offers a proven method of designing controllers that are robust to such uncertainty. However, with conventional H_∞ control, the performance of the resulting system can often be unsatisfactory. The following subsections will provide an overview

of the flight dynamics of an aircraft and introduce the concept of H_∞ control and loop shaping - a technique that allows the designer to 'shape' the response of a system, and thus improve performance.

Flight Dynamics

The dynamics of an aircraft in flight can be described by the rotational moments around its centre of gravity (CG) in Cartesian space. These are shown in Figure 5.16 where:

- **L** is the *rolling* moment of the aircraft.
- **M** is the *pitching* moment of the aircraft.
- **N** is the *yawing* moment of the aircraft.

These flight dynamics can be separated into *longitudinal* motions, in which the wings remain level (e.g. pitch), and *lateral* motions such as roll and yaw.

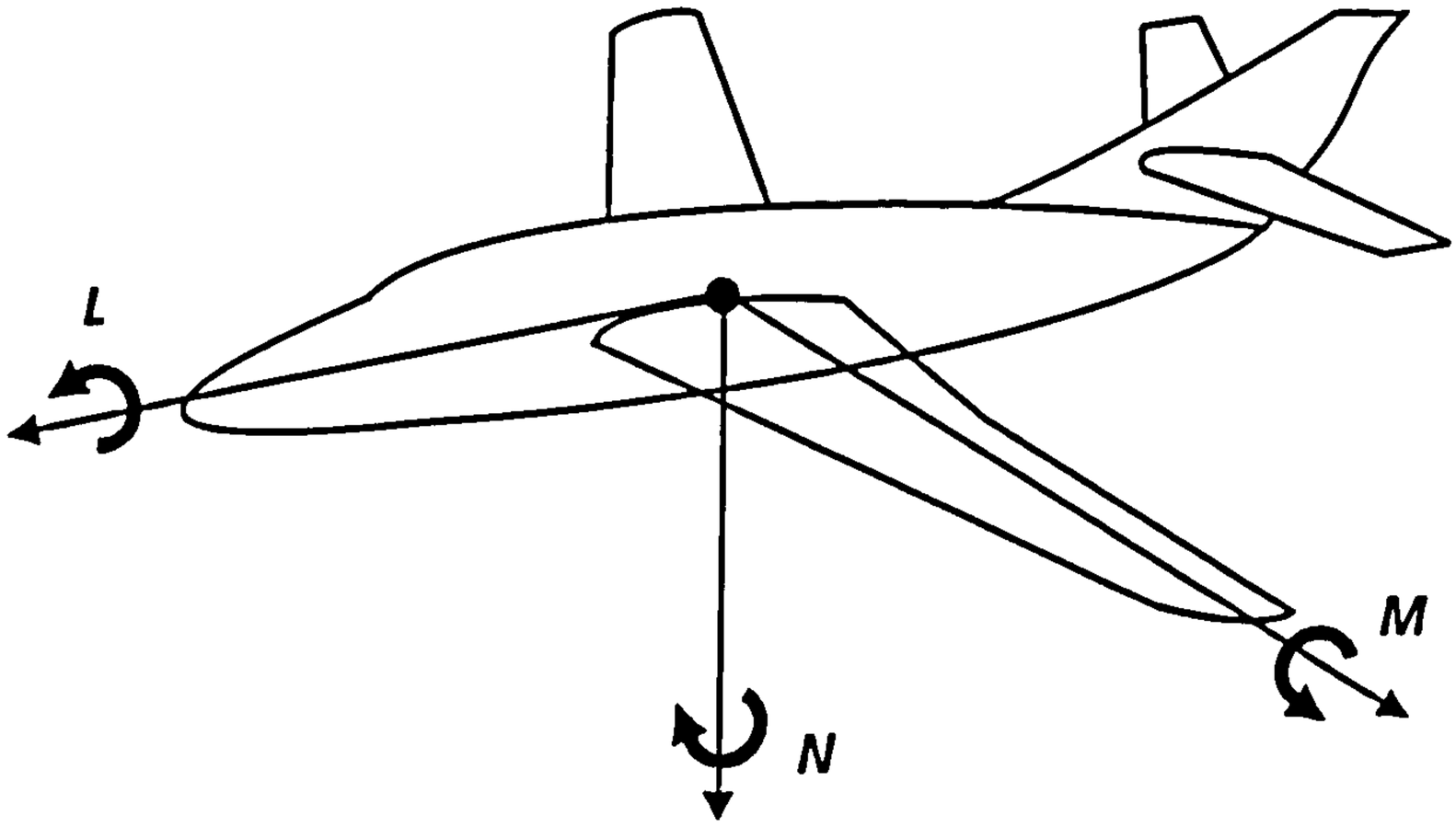


Figure 5.16: The Rotational Moments of an Aircraft

The equations of motion describing these flight dynamics are non-linear; however, by applying the *small disturbance theory* (Nelson 1998), a linearised

model can be found. This linearisation process will only give a good result in cases where the motion of the aircraft can be fully described by small deviations about a steady flight condition (such as in the flight of large commercial aircraft), and therefore should not be used in cases where large amplitude motions are likely to occur.

H_∞ Loop Shaping Controller Design

H_∞ control theory was originally proposed by Zames (1981) to address the problem of uncertainty in the modelling of disturbances and plants and was further developed by Glover and Doyle (1988). H_∞ control theory provides a general framework for the design of optimal controllers, where optimal in this context refers to the minimisation of the H_∞ norm.

A drawback of robust stabilisation using H_∞ control is the inability of the designer to specify performance requirements (Skogestad and Postlethwaite 1996), which can result in compensated systems that, whilst robust, perform unsatisfactorily. To overcome this limitation, McFarlane and Glover (1990) proposed using pre- and post-compensators to 'shape' the open-loop response of the plant (see Figure 5.17), and then applying robust stabilisation. Selecting the weighting matrices for the pre- and post-compensators is typically challenging since these choices govern the performance of the resulting system.

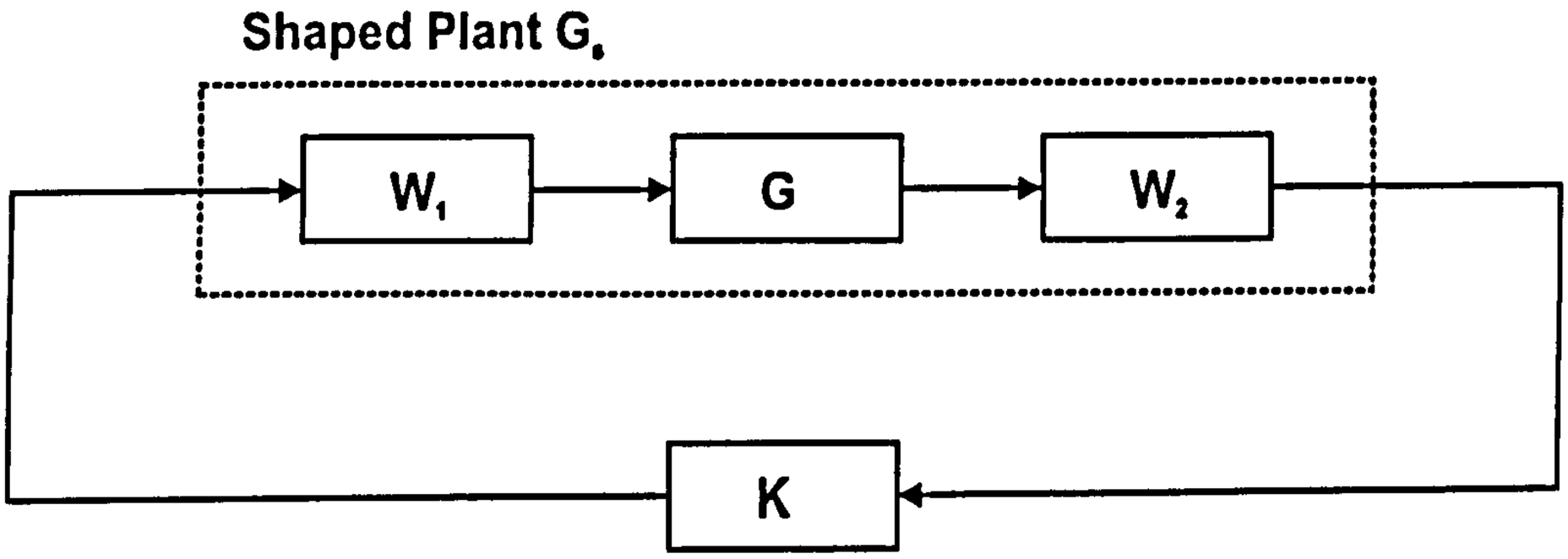


Figure 5.17: The Shaped and Compensated Plant

The general proceedure for designing an H_∞ controller using loop shaping is (Skogestad and Postlethwaite 1996):

1. Scale the inputs and outputs of the system to improve the condition of the design problem.

2. Select the weighting matrices for the pre- and post-compensators and form the generalised shaped plant model (see Figure 5.17). As mentioned above, this is typically challenging due the influence this stage has on the performance of the final compensated system.

3. Robustly stabilise the shaped plant, $G_s = W_2GW_1$, using standard H_∞ control techniques such as those described in Skogestad and Postlethwaite (1996).

4. Test the performance and stability of the compensated system and, if the requirements are not met, alter the weighting matrices formed in step 2.

Optimal Controller Design

Evolutionary optimisation methods have previously been used in finding both optimal controller parameters and optimal controller structures (Schroder 1998). This Section aims to find an optimal H_∞ loop shaping controller for the lateral stability control of a Boeing 747 aircraft[10]. This is a multivariable system with two inputs (the control signals for the aileron and rudder) and two outputs (the roll and sideslip angles), and can be represented by the following transfer function matrix[11]:

---

[10]For a thorough investigation into the design of an H_∞ lateral stability controller the interested reader is directed to (Giacoman Zarzar 2007).

[11]This can be shown to be a minimal realisation of the system using the *minreal* command in MATLAB.

$$\mathbf{G} = \begin{pmatrix} \frac{0.1845s^2+0.04795s+0.1995}{s^4+0.6807s^3+1.049s^2+0.3373s-0.001979} & \frac{0.06591s^2-0.12s-0.5158}{s^4+0.6807s^3+1.049s^2+0.3373s-0.001979} \\ \frac{-0.01448s^2-0.01962s+0.001359}{s^4+0.6807s^3+1.049s^2+0.3373s-0.001979} & \frac{0.005334s^3+0.4377s^2+0.1884s-0.00432}{s^4+0.6807s^3+1.049s^2+0.3373s-0.001979} \end{pmatrix}$$

In this Section, the Grid-enabled framework for evolutionary optimisation introduced in this Chapter is used to determine the optimal weights (with respect to the performance and stability requirements of the system) for the pre- and post-compensators in the H_∞ loop shaping design process outlined previously. The pre- and post-compensators have the following structures:

$$\mathbf{W_1} = \begin{pmatrix} \frac{s+a}{s} & 0 \\ 0 & \frac{s+b}{s} \end{pmatrix}$$

$$\mathbf{W_2} = \begin{pmatrix} c & 0 \\ 0 & d \end{pmatrix}$$

with the order of these compensators being determined by the order of the plant.

This H_∞ loop shaping controller design process can be formulated as a multi-objective optimisation problem, where each performance requirement is treated as a separate objective, and thus solved using a multi-objective evolutionary algorithm. The decision variables in this optimisation procedure are the weights, a, b, c, d, in the compensators. However, this controller design problem is computationally expensive since, for every candidate solution, an H_∞ controller has to be synthesized and the response of the compensated system obtained by computer simulation. Evaluation of a single candidate solution for this problem took in the order of 5.5 seconds on a Intel Pentium 4 based PC running at 3.0GHz.

A set of performance requirements arising from domain specific knowledge about the problem have been specified for the response of the compensated system

(see Table 5.6). Some of these requirements are hard constraints and others are simply desired goals. Several of the performance requirements for this controller are in competition which makes achieving all the goals difficult.

Requirements	Type
Minimise the Overshoot in response to a step input	Goal (Overshoot < 5%)
Minimise the Rise Time	Goal (T_r < 3 seconds)
Minimise the Settling Time	Goal (T_s < 4 seconds)
Prevent Aileron actuator saturation	Constraint (Aileron deflection < 0.349 radians)
Prevent Rudder actuator saturation	Constraint (Rudder deflection < 0.52 radians)
Controller must be robust to 30% multiplicitive uncertainty	Constraint

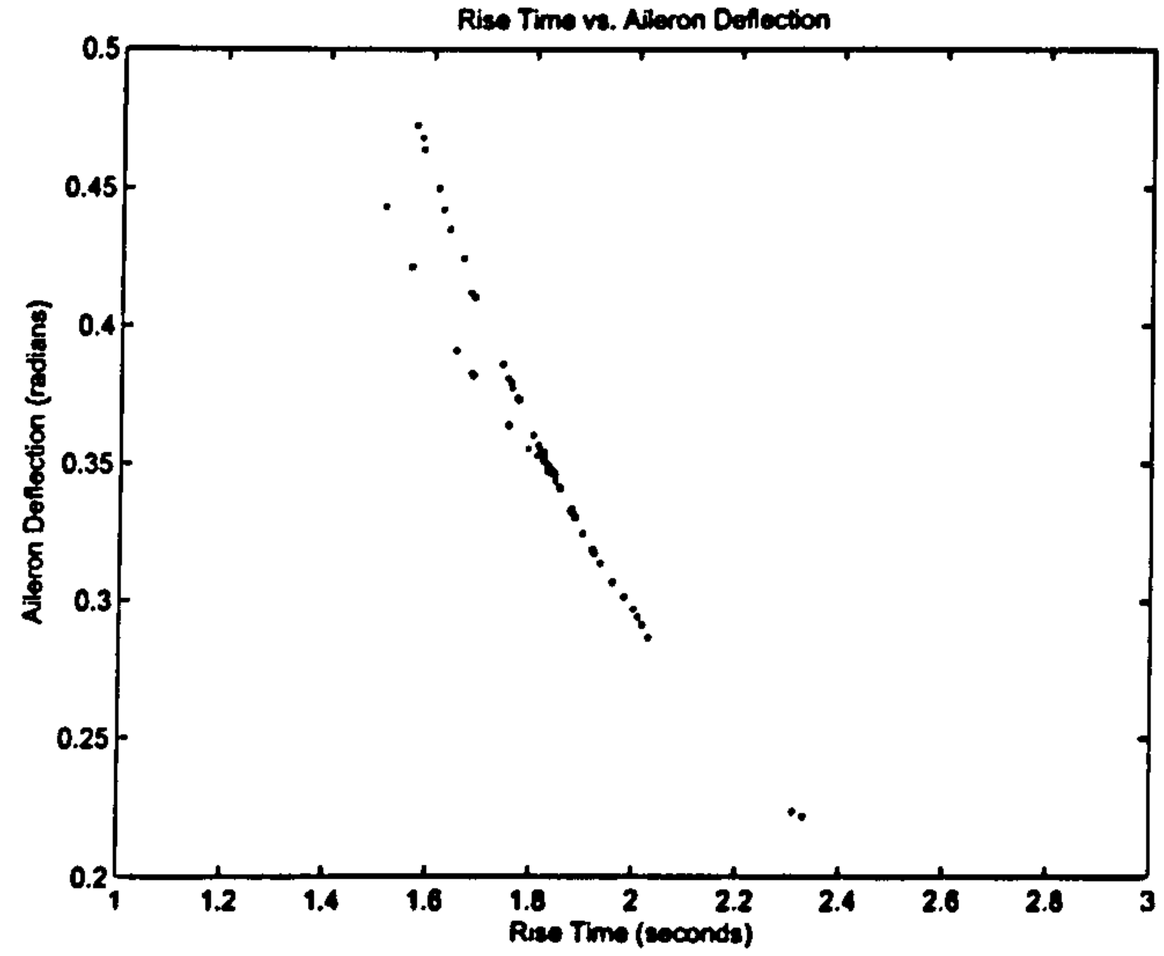
Table 5.6: Performance Requirements for the H_∞ Controller Design Problem

Evolutionary Algorithm Implementation

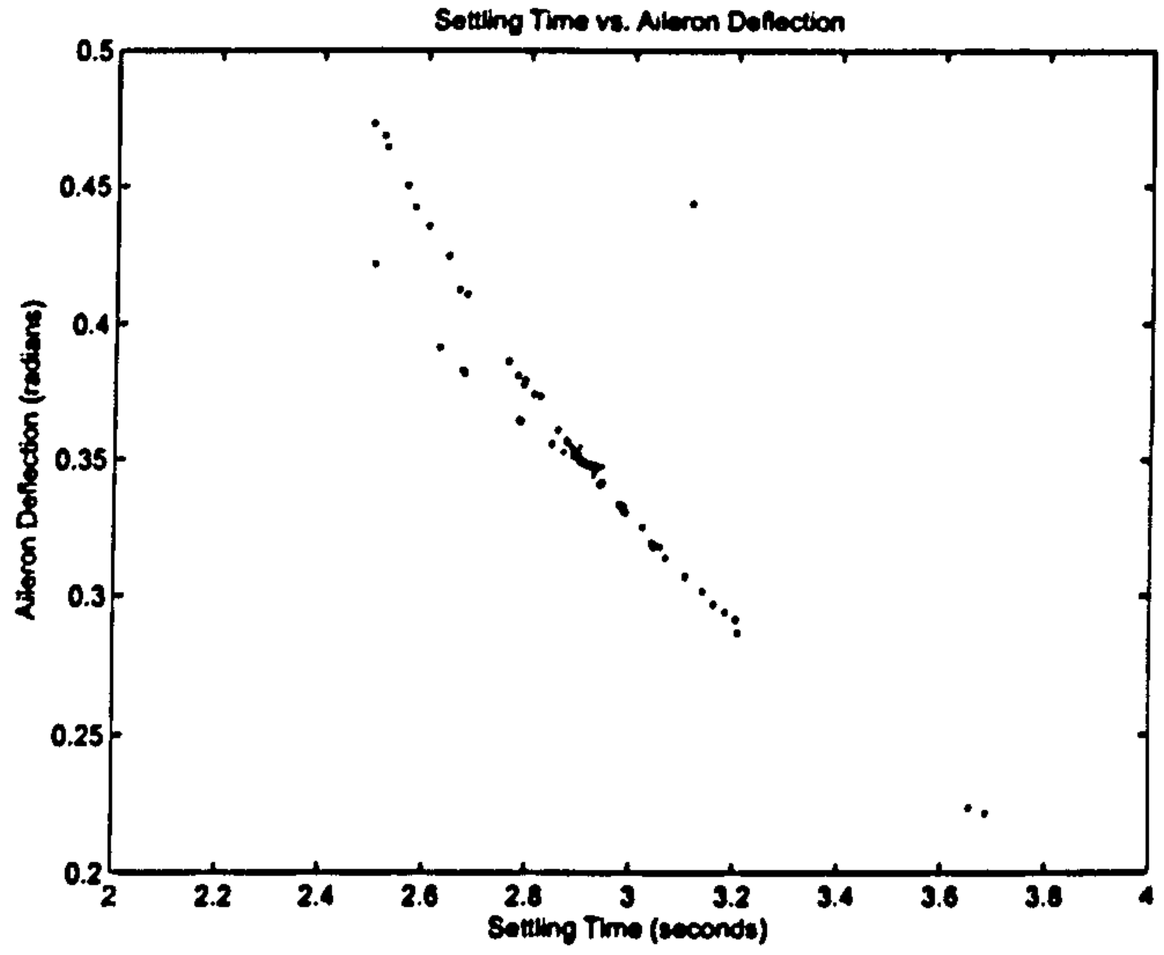
The Grid-enabled optimisation framework propose in this Chapter was used here in a multi-objective genetic algorithm architecture, with Fonseca and Fleming's (1998) modified Pareto ranking scheme used to incorporate the preference information shown in Table 5.6 into the search. As in Section 5.5.1, a real-valued decision variable representation was used, with selection being performed using Stochastic Universal Sampling (Baker 1987) and variation introduced into the population using extended intermediate recombination and BGA mutation (Mühlenbein and Schlierkamp-Voosen 1993). The population size was 100 individuals, and the MOEA was run for 100 generations.

Results

It can be seen from Figure 5.18 that there is a strong trade-off between the control signal for the aileron actuator and the other requirements in Table 5.6 (such as



(a)



(b)

Figure 5.18: Trade-offs Between Objectives and the Aileron Control Signals

the the rise time and settling time). This shows that the response of the final compensated system is limited by the maximum range of the aileron actuator.

Table 5.7 shows the achieved performance of the compensated system and Figure 5.19 shows the response of the system to a 0.1 radians step change in the aileron control signal. It can be seen from Table 5.7 that all the performance

requirements specified in Table 5.6 are satisfied, and significant improvements over all the goal values have been achieved.

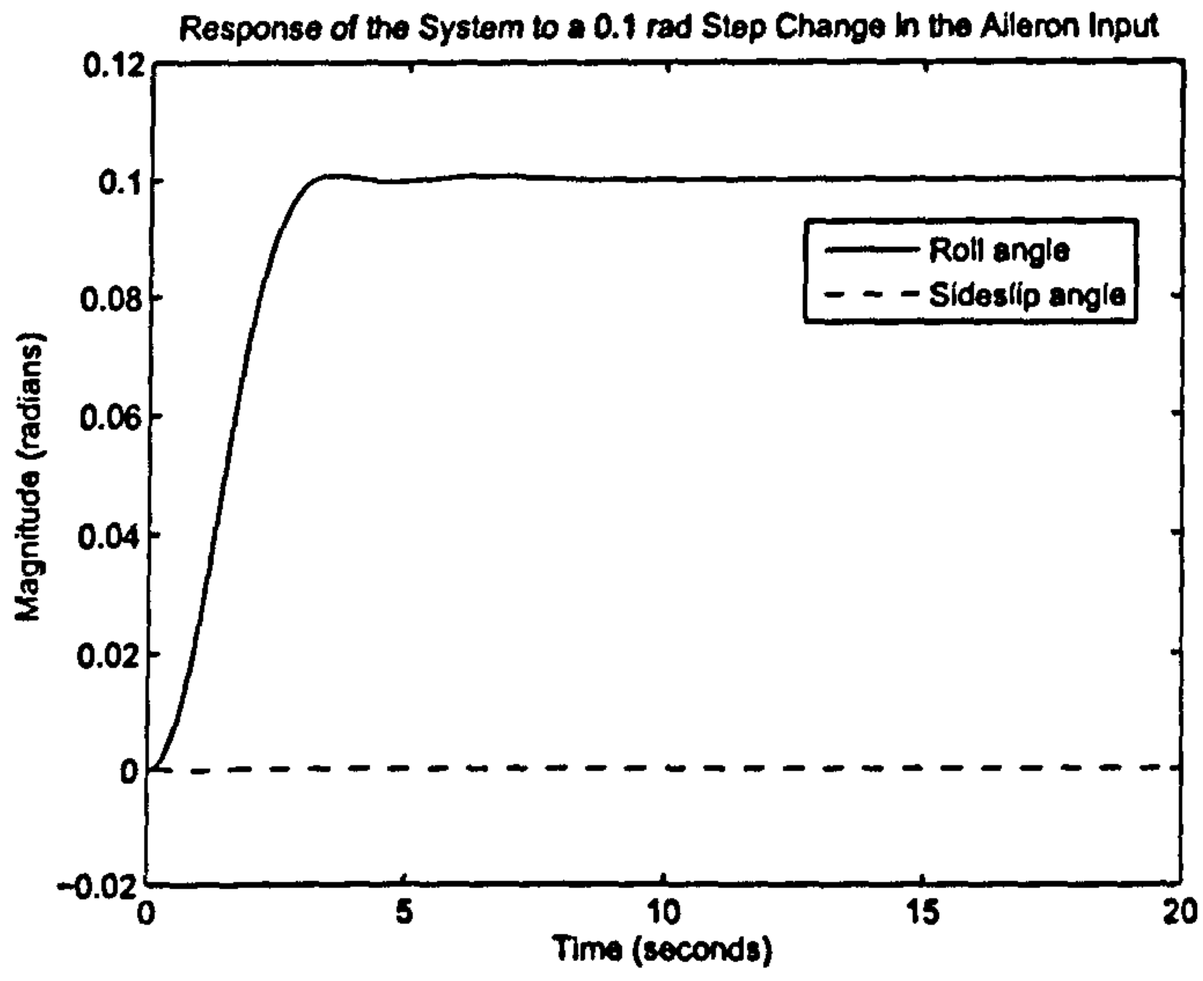
Goals	Achieved Value
Minimise the Overshoot in response to a step input (Goal: Overshoot < 5%)	0.87%
Minimise the Rise Time (Goal: T_r < 3 seconds)	1.83 seconds
Minimise the Settling Time (Goal: T_s < 4 seconds)	2.9 seconds
Prevent Aileron actuator saturation (Constraint: Aileron deflection < 0.349 radians)	0.3488 radians
Prevent Rudder actuator saturation (Constraint: Rudder deflection < 0.52 radians)	0.0188 radians
Controller must be robust to 30% multiplicitive uncertainty (Constraint)	Robust to 35.9% multiplicitive uncertainty

Table 5.7: Achieved Performance of the Compensated System

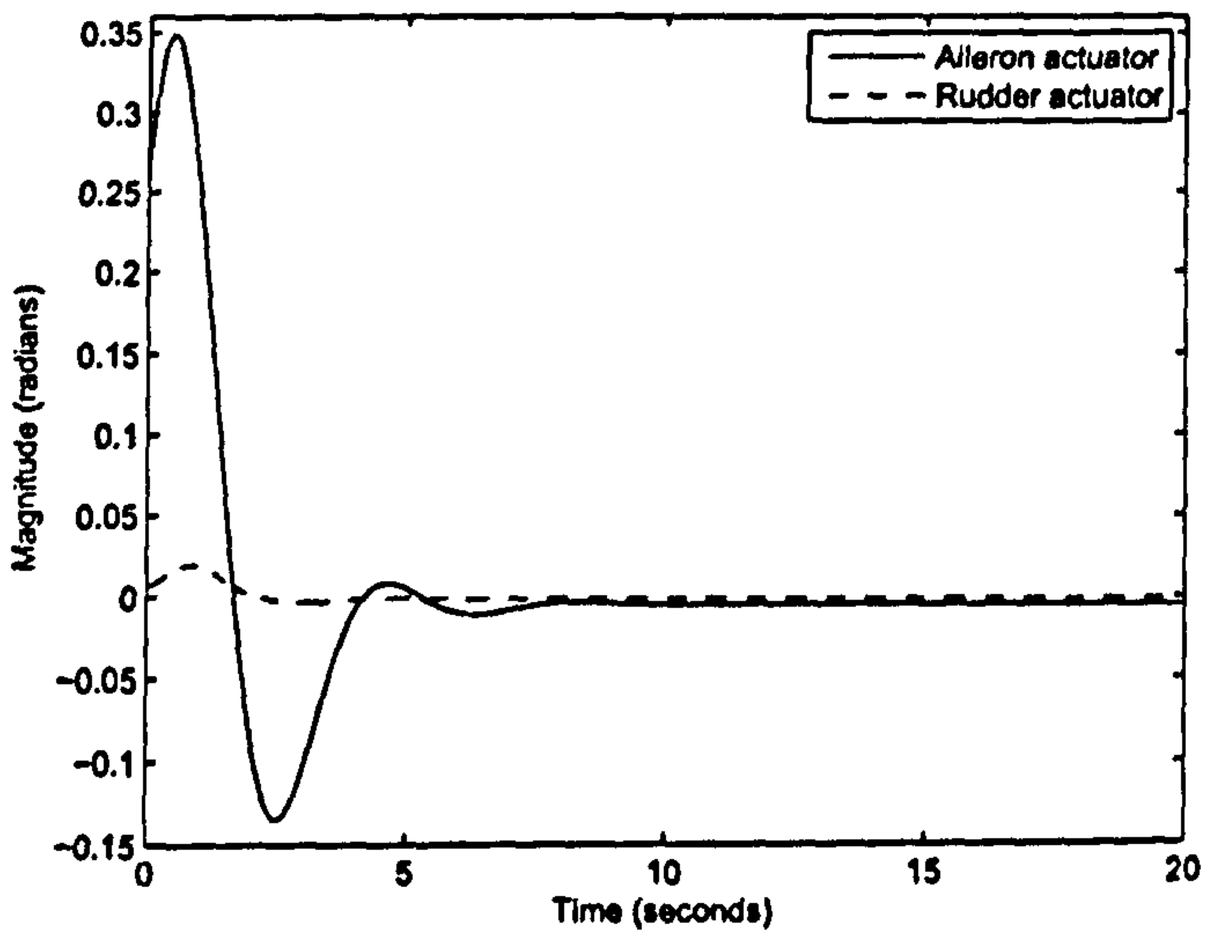
Table 5.8 shows a representative set of execution times from the evolutionary multi-objective optimisation of the controller design problem presented in this Section. These times are averaged over 5 runs for both the results obtained from a single workstation and the results obtained using the resources of the White Rose Grid (see Section 4.3). As Table 5.8 shows, the Grid-enabled optimisation framework presented in this Chapter has significantly reduced the time taken to optimise the performance of the compensated system.

Single Workstation		Computational Grid	
50 generations	100 generations	50 generations	100 generations
28637 seconds	56698 seconds	8251 seconds	12582 seconds

Table 5.8: Execution Times for the Optimisation of an H_∞ Controller Design Problem



(a)



(b)

Figure 5.19: Step Response of the Compensated System to Aileron Deflection

Discussion

Previous work (Giacoman Zarzar 2007) has shown that the evolutionary multi-objective optimisation approach to H_∞ loop shaping controller design can produce solutions that significantly outperform those obtained using conventional loop shaping. This is because the use of evolutionary multi-objective optimisation techniques allows the designer to explore a much larger region of the search space than would otherwise be possible. The downside to this approach is the time taken by the optimiser to reach a solution due to both the population based, iterative nature of EMO techniques, and the computationally expensive evaluation of candidate solutions. The example presented in this Section has shown that using the Grid-enabled optimisation framework introduced in this Chapter can significantly reduce this execution time, thus mitigating the main drawback of this approach.

5.6 Summary

This Chapter has introduced a Grid-enabled framework for evolutionary optimisation. Firstly, a review of the potential population topologies for parallel evolutionary computation was undertaken and an analysis of previous related work was performed. It was found that much of the research effort into parallel evolutionary computation has focussed on the single-objective case, and what little research there has been into multi-objective parallel EAs (such as that by Van Veldhuizen *et al.* (2003) and Deb *et al.* (2003)) has not attempted to take advantage of the recent paradigm of Grid computing.

Section 5.3 compared the performance of the two main paradigms of parallelism in evolutionary computation on a variety of single and multi-objective test problems taken from the literature. It was found that the different parallelisation strategies suited different types of problems, with the island model performing

best in unimodal problems and the globally parallel algorithm performing best in multimodal search spaces. However, the choices for the extra parameters in the island model EA - especially the choice of the number of subpopulations - were found to be difficult. Although guidelines have been proposed in the literature for setting some of the parameters (see, for example, Cantú-Paz (1999)), it was found that the optimal number of subpopulations was problem dependent. It was therefore chosen to implement the Grid-enabled framework proposed in this Chapter using a globally parallel model, since the choice of population topology and number of subpopulations in the island model may otherwise be determined by the configuration of the Grid resources and thus not be optimal. The optimisation framework introduced here was also implemented in a service-oriented architecture, unlike many of the other Grid-based evolutionary computation environments which rely on monolithic architectures. This enabled the framework to be very flexible, whilst also being well suited to the kind of master-worker parallelism implemented here.

The Grid-enabled optimisation framework proposed in this Chapter is primarily suited to computationally expensive objective function evaluations, such as those performed by computer simulation. This is due to the communication overheads inherent in evaluating the objective functions in a distributed manner. Figure 5.20 shows the scale of evaluation functions that the proposed framework is effective for. In this experiment each generation consisted of 50 individuals, and the computational complexity of the objective function evaluation was varied. Multiple runs of the experiment were performed, and the results averaged.

As can be seen from Figure 5.20, using this Grid-enabled framework for optimisation of computationally trivial objective functions will result in a decrease in performance when compared to a sequential EA. However, for objective functions that take over 0.5 seconds to evaluate a single individual, substantial savings can be achieved in the total execution time of the algorithm.

It is expected that further research and development into Grid middleware,

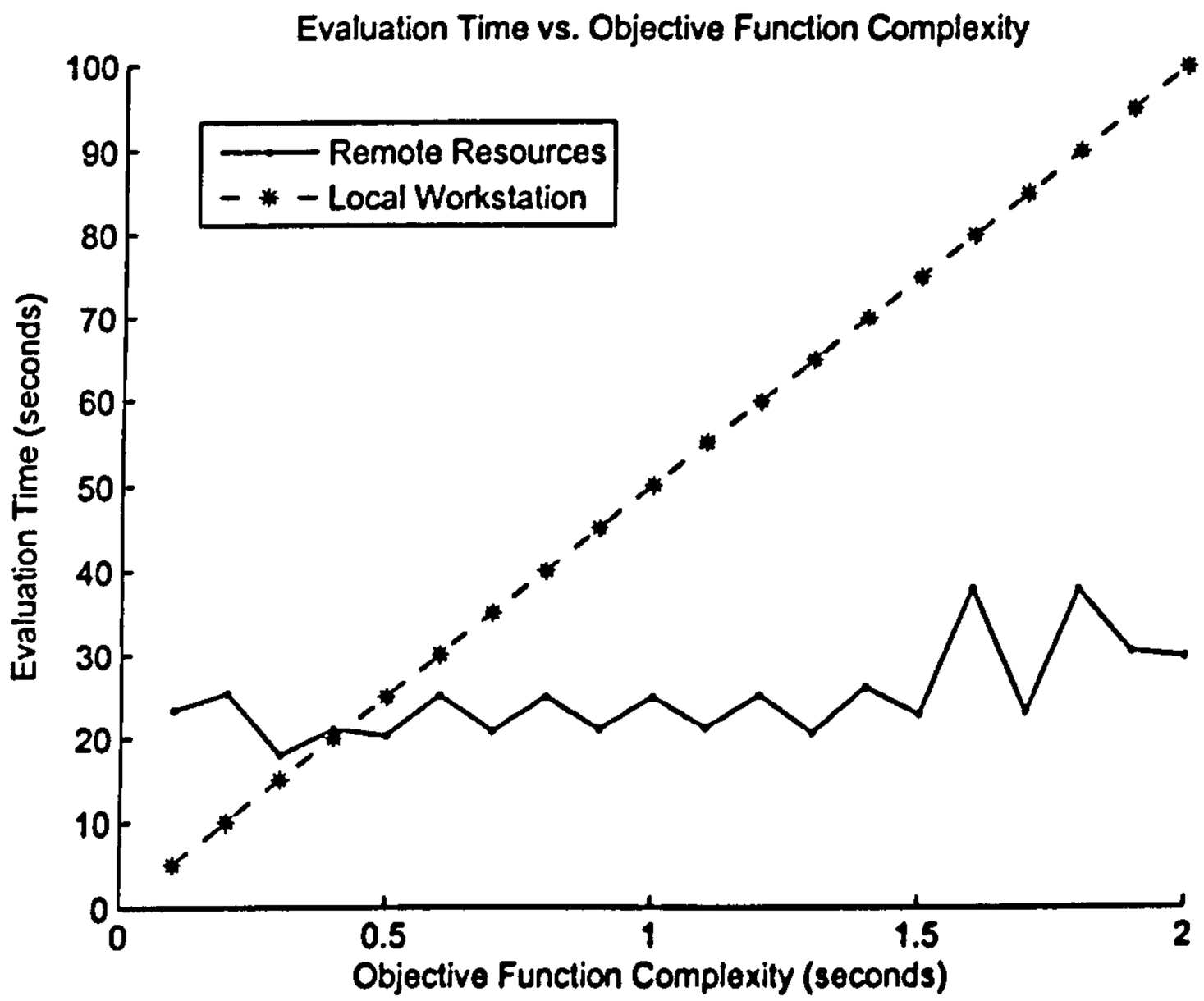


Figure 5.20: The Scale of Objective Functions the Proposed Framework is Effective For

resource management systems, and network infrastructure will result in reductions in the communication overheads present in the proposed framework. This will enable this framework to provide increased performance for less computationally expensive problems. However, this framework is not intended to replace sequential EAs in cases where the performance of sequential EAs is satisfactory.

This Grid-enabled optimisation framework was used to optimise two computationally expensive real-world problems under investigation in the author's research group. The first problem involved the single-objective optimisation of maintenance scheduling strategies for Rolls-Royce aero-engines. By using the Grid-enabled framework introduced in this Chapter, substantial speed-ups in the execution time of the algorithm were achieved. These speed-ups enable the application of evolutionary computation techniques to larger scale problems, such as those with larger fleets of aircraft, as well as providing a decision maker with the ability to run the optimiser multiple times with alternative cost models and failure distributions so as to obtain a greater understanding of the problem space.

The second real-world problem involved the multi-objective optimisation of the performance of an H_∞ loop shaping controller for the lateral stability of a Boeing 747 aircraft. Conventional H_∞ controllers can often result in systems that, whilst robust, are unsatisfactory in performance. One way to overcome this problem is to use loop shaping to improve the performance of the resulting system. However, the choice of weighting matrices for the pre- and post-compensators can be a challenge. Using evolutionary multi-objective optimisation techniques allows the control systems designer to explore a much larger region of the search space than would otherwise be possible, and potentially discover solutions that would not have been considered. The downside to this approach is the time taken by the optimiser to reach a solution. The Grid-enabled framework presented in this Chapter was used to achieve a considerable reduction in the execution time of the optimiser, and can therefore provide an engineer with a powerful tool in the

design of control systems. Although this example focussed on the design of an H_∞ controller, the framework is equally applicable to the design of other control systems, such as PID controllers, where the evaluation of the performance of the plant requires computationally expensive computer simulation.

The results from the Grid-enabled optimisation of the real-world examples presented in this Chapter show that substantial speed-ups in the execution time of the optimiser can be achieved. The results also show that the potential speed-up achieved by a problem increases with the computational complexity of the objective function evaluations. This is partly due to the high overall utilisation of the available Grid resources. Therefore, as the complexity of the problem increases, the amount of time spent waiting for the jobs to run becomes less significant and thus the potential speed-up increases. Another way to increase the performance of the proposed optimisation framework would be to negotiate Service Level Agreements (SLAs) with the resource providers to guarantee a minimum level of service for the objective function evaluations.

Chapter 6

Computational Steering of Evolutionary Multi-Objective Optimisation

6.1 Introduction

Decision making in engineering design can frequently be aided by using evolutionary algorithms to solve multi-objective optimisation problems. Typically these multi-objective evolutionary algorithms are run non-interactively, with the decision maker (DM) setting the initial parameters of the algorithm and then executing it for a set number of generations. During this execution process, which can often take hours or days to complete, user interaction, if any, is limited to the occasional plotting of the intermediate solutions and the possible termination of the algorithm if it appears to have failed (for example, if the algorithm fails to converge). When the optimisation process has completed, the final solutions produced by the algorithm are assessed and, if the results are not satisfactory, the parameters of the algorithm are altered and it is run again. This process of repeated execution of a multi-objective evolutionary algorithm leads to an

inefficient use of resources, and also to possibly inferior solutions.

As the process of setting the initial parameters of the algorithm can be difficult, especially if little is known about the problem domain, the re-execution of the algorithm with altered parameters is common. Unfortunately the evolutionary computation community is still some way from possessing anything more useful than 'rules-of-thumb' when it comes to setting these initial parameters (Bullock *et al.* 2002). One potential solution to this problem is to allow the decision maker to interact with the optimisation routine during execution. This is known as computational steering, and may be as simple as allowing the decision maker to monitor the values of some parameters in the optimisation process and, if necessary, to adjust others. In this way the decision maker can influence the quality of the solutions produced by the optimisation process.

Research into interactive evolutionary computation has mostly focussed on partial or complete human evaluation of solutions produced by an evolutionary search (Parmee 2002). This technique has mainly been applied to those problems that rely on a subjective evaluation of the fitness of a particular solution, such as the evolutionary design of computer graphics (Sims 1991) or music (Biles 2003). There is little research into the computational steering of evolutionary computation at run-time, and the research that exists (Bullock *et al.* 2002) focuses solely on the single-objective case.

This Chapter aims to show that computational steering of a multi-objective evolutionary algorithm can provide improved performance in both the execution speed of the algorithm and in the quality of the solutions that the algorithm produces. Section 6.2 briefly outlines the ideas behind computational steering, and then, in Section 6.3, the role of the decision maker in engineering design and ways of incorporating the DM's preferences into the optimisation process are discussed. Section 6.4 describes the implementation of the proposed steering system, and in Section 6.5 it is applied to a many-objective aircraft controller design problem. Section 6.6 draws some conclusions about the use of

computational steering in evolutionary multi-objective optimisation.

6.2 Computational Steering

Computational steering is defined as an approach that improves the integration of simulation and visualisation in the computational process, allowing the engineer or scientist to control the succession of steps required to solve engineering and computational science problems (Johnson *et al.* 1999). The desire to interact with their simulations is nothing new for engineers and scientists. In 1987, the Visualization in Scientific Computing Workshop reported:

> "Scientists not only want to analyse the data that results from super-computations; they also want to interpret what is happening to the data during super-computations. Researchers want to *steer* calculations in close to real time; they want to be able to change parameters, resolution, or representation, and see the effects. They want to drive the scientific discovery process; they want to *interact* with their data." (McCormick *et al.* 1987)

Currently, the majority of computational steering systems are applied to large simulations involving compute-intensive models. One area of application is in computational fluid dynamics (CFD) where lattice-Boltzmann simulations can be used to understand the behaviour of meso-scale fluid systems (Chin *et al.* 2003). Examples of such systems occur in everyday life, for example in detergents, milk and blood. In this application, computational steering is used to overcome the limitations of the simulate-then-analyse approach, such as having a fixed number of time steps in the simulation. Another area of application of computational steering is in medical science. The SCIRun computational steering system (SCIRun 2006) has been used in EEG simulation and visualisation (Johnson

et al. 2004) and to plan operations by performing interactive visualisation of tumours. This planning and simulation is of great benefit to surgeons as it allows them to practice the operations they are to perform.

The visualisation of the intermediate results of the computational process is extremely important. It must allow the engineer or scientist to efficiently extract the relevant information from the data (Parker *et al.* 1997), so as to be able to make an informed choice about which aspects of the process to adjust. The complexity of the visualisation should be able to be tailored to the hardware available to the user (for example, a laptop computer, Personal Digital Assistant (PDA), etc.) (Brooke *et al.* 2003).

For an application to be 'steerable' it must have a point in the control loop of the program where it can be interrupted and steering tasks can be performed. This point is commonly termed a break-point and should allow some or all of the following tasks to occur (Brooke *et al.* 2003):

1. The retrieval of the current parameter set.

2. The alteration of one or more parameters.

3. The retrieval of the current results set (which can then be passed to a visualisation routine).

4. The taking of a 'snap-shot' of the application (often referred to as a checkpoint) to allow the system to be restarted from this point.

The first three tasks in this list form the basis of the computational steering process (i.e. the alteration of simulation parameters in response to the results currently being produced by the simulation). The support of checkpointing in large-scale complex simulations may be difficult, however, and this decision should be left up to the application user.

6.3 Decision Making in Engineering Design

The main role of a decision maker in evolutionary multi-objective optimisation is usually to select a single solution from the potentially infinite Pareto-optimal solution set, according to some criteria. In practice the DM is usually only interested in a sub-set of the trade-off surface, thus there is little or no benefit in representing parts of the trade-off surface that lie outside this region of interest. Allowing the DM to focus the search on relevant areas of the solution space increases the efficiency of the optimisation process and reduces the amount of irrelevant information that the decision maker has to consider (Fleming *et al.* 2005), thus preventing them from becoming overwhelmed. This is especially important in problems with many conflicting objectives, since the increased scale of these problems means that the global trade-off surface will contain many Pareto-optimal solutions, most of which may not be of interest to a decision maker (Purshouse 2003).

As mentioned in Section 2.4.5 earlier, DM preferences can be incorporated into the optimisation process in three ways: *a posteriori*, *a priori*, and *progressively*. *A posteriori* methods of preference articulation involve the DM selecting a compromise solution from the global set of Pareto-optimal solutions, whilst *a priori* and progressive preference articulation methods aim to achieve a good representation of the trade-off surface in the DM's region of interest. They do this by concentrating the optimiser on a sub-set of the global trade-off surface.

In *a priori* articulation of preferences the DM expresses their preferences before the start of the optimisation process. However, often the DM may not be sure of their preferences prior to optimisation, and by stating preferences *a priori* the DM may not investigate some areas of the search space that merit attention. A better method is progressive articulation of preferences, where the DM can express preferences during the search and thus incorporate information that becomes available during the search process.

Adra *et al.* (2007) reviewed several recent and established methods for progressive articulation of preferences, including Fonseca and Fleming's (1998) preferability operator (see Section 2.4.5 earlier), and interactive versions of Branke *et al.*'s (2001) guided dominance scheme, Branke and Deb's (2004) biased crowding technique, and the ϵ-dominance concept (Suganathan *et al.* 2005). They concluded that Fonseca and Fleming's (1998) preferability operator provided the decision maker with a user-friendly technique that allowed preferences to be directly expressed in terms of values for the objectives (Adra *et al.* 2007). This is in contrast to the other techniques surveyed which required the specification of parameters that were not easily translatable to objective values. These methods of progressive preference articulation represent a limited form of computational steering; however, they do not allow a DM to alter any of the underlying optimisation parameters that effect the behaviour of the algorithm.

The ability for the DM to computationally steer the optimisation routine is desirable due to the large amount of time such optimisation routines often take to complete. By giving the DM the ability to observe the progress of the optimisation process and to alter the parameters as the algorithm runs, the DM can act to improve the quality of the solutions produced by this optimisation process. For example, if the optimisation routine is struggling to find solutions of interest to the DM, then the DM could alter the mutation rate, change the bounds on the decision variables, or express design preferences to try to improve the quality of the solutions produced by the algorithm.

6.4 Implementation of a Computational Steering System for Multi-Objective Optimisation

6.4.1 Steering of the Multi-Objective Evolutionary Algorithm

To implement computational steering for an application, the application must first be 'steerable'. In Section 6.2 it was noted that for computational steering to be applied to an application there must be a suitable break-point in its control loop where the program can be interrupted and the steering operations performed.

An evolutionary optimisation routine contains an ideal break-point due to its iterative nature. This point occurs between the generations of the algorithm, and provides a convenient place to retrieve the current candidate solutions and parameter set, and to alter those parameters that are appropriate (see Figure. 6.1). However, if there are no steering tasks to be performed then the optimisation routine will continue to the next generation.

```
Procedure SteeringEA:
    Initialise()
    Evaluate()
    While not finished:
        Selection()
        Recombination()
        Mutation()
        Evaluate()

        End
End
```

Break-point - perform steering tasks

Figure 6.1: Pseudo-Code Illustrating the Positioning of a Break-Point in the Control Loop of the Optimiser

Because the next generation of candidate solutions produced by the optimisation routine only relies on the current generation of candidate solutions and the current parameter set it is also simple to implement a checkpointing system. This will take the form of a 'snap-shot' of the state of the algorithm, taken at the break-point, consisting of the current parameter set and the current generation of candidate solutions. If the optimisation routine is to be restored from this point it is a simple matter to reload this 'snap-shot' and continue the optimisation process.

The actual computational steering of the optimisation process can be achieved in two ways. The first of these is by adjusting the parameters of the algorithm. These parameters control the behaviour of the algorithm and can affect both the rate of convergence and the quality of the solutions produced. For example, reducing the exploratory effects of mutation in the algorithm by lowering the mutation rate will reduce the amount of new genetic material coming in to the population in each new generation, and thus increase convergence. However this increase in the rate of convergence will come at the risk of converging to a local optimum.

Some other parameters that can be adjusted in the proposed steering system are the upper and lower bounds on the decision variables, the population size (either by increasing the number of immigrants or increasing the number of solutions produced by selection), and the fitness assignment method. These parameters all affect the behaviour of the algorithm in different ways. For instance, tightening or loosening the bounds on the decision variables allows the engineer to focus or widen the search in decision space, while increasing the number of immigrants in the population can force the algorithm out of local optima because it introduces new genetic material.

The engineer can also alter the probability of a solution being carried over to the next generation by changing the method by which fitness is assigned. For example, if an exponential fitness assignment method is used, then the highest

ranked solution will form a proportionally larger part of the next generation compared to a linear fitness assignment method.

The second way of steering the optimisation process is to use progressive preference articulation (see Section 6.3) to alter the goals and priorities for the objectives, and thus affect the areas of the search space that the algorithm focuses on. The areas of the search space that the algorithm focuses on are defined by the preferences specified by the DM, and the algorithm focuses on this region of interest by assigning a higher rank to those solutions that are in this region. Once a satisfactory value has been achieved for one of the objectives, the objective in question can be constrained to be at least as good as that value. All the potential solutions that do not meet this criterion are ranked worse than those that do, and therefore the algorithm is steered away from values that violate that constraint. Reducing the region of interest in this way can accelerate the optimisation process and improve the quality of useful solutions.

One possible alternative to computational steering for tuning key parameters in evolutionary computation is the concept of Nested Evolution. This was introduced for a differential evolution (DE) algorithm in (Babu *et al.* 2004), and consists of an inner optimisation loop and outer optimisation loop. The inner loop aims to solve the problem, whilst the outer loop optimises the key parameters of the inner loop (with the objective of minimising the number of generation the inner loop runs for). The main drawback of this method is that, as the inner loop is a population based evolutionary optimiser, a large number of function evaluations may be needed to tune the key parameters in the inner loop. Many non-trivial, real-world problems require the use of computationally intensive computer simulations to evaluate the candidate solutions produced by the optimiser, and it is therefore desirable to keep the number of function evaluations to a minimum. For this reason it was felt that computational steering was a better choice than nested evolution.

6.4.2 Visualisation

As mentioned in Section 6.2, visualisation is a key component of computational steering. The visualisation method of any computational steering system must be able to present the user with enough relevant information for the user to guide the process. Therefore, the visualisation method for the intermediate results of the multi-objective evolutionary algorithm must be able to display high-dimensional data sets, as we are dealing with many objectives.

The visualisation of high dimensional data sets in an intuitive manner is extremely difficult. While scatter diagrams provide a fundamental tool for visualisation of lower dimensional data - allowing the eye to see such features as clustering, outliers and non-linearities - they do not generalise easily to more than three dimensions (Wegman 1990). Several methods have been proposed to solve this problem (ATKOSoft 1997), but this Section will focus on four of the most commonly used high-dimensional visualisation techniques.

Scatter plot matrices (see Figure 6.2) are a commonly used technique in the visualisation of high dimensional data sets. They provide a visualisation technique that facilitates rapid scanning of many dimensions; however, discovery of high dimensional patterns can be complicated by the disconnected representation of multiple aspects of the same point in high dimensional space (Carr *et al.* 1986). The representational complexity of these scatter plot matrices is high ($O(n^2)$), because they project n dimensions onto $n \times (n-1)$ scatter plots. This means that this technique does not scale well to large numbers of variables. This high representational complexity also means that this technique will be unsuitable for on a device with limited screen size, for example that of a PDA.

Star Plots (Chambers *et al.* 1983) are an iconic representation of multidimensional data, used mainly to illustrate trends in that data. Each point in multidimensional space is represented by a star and the points can either be examined individually or used to identify clusters of points with similar features.

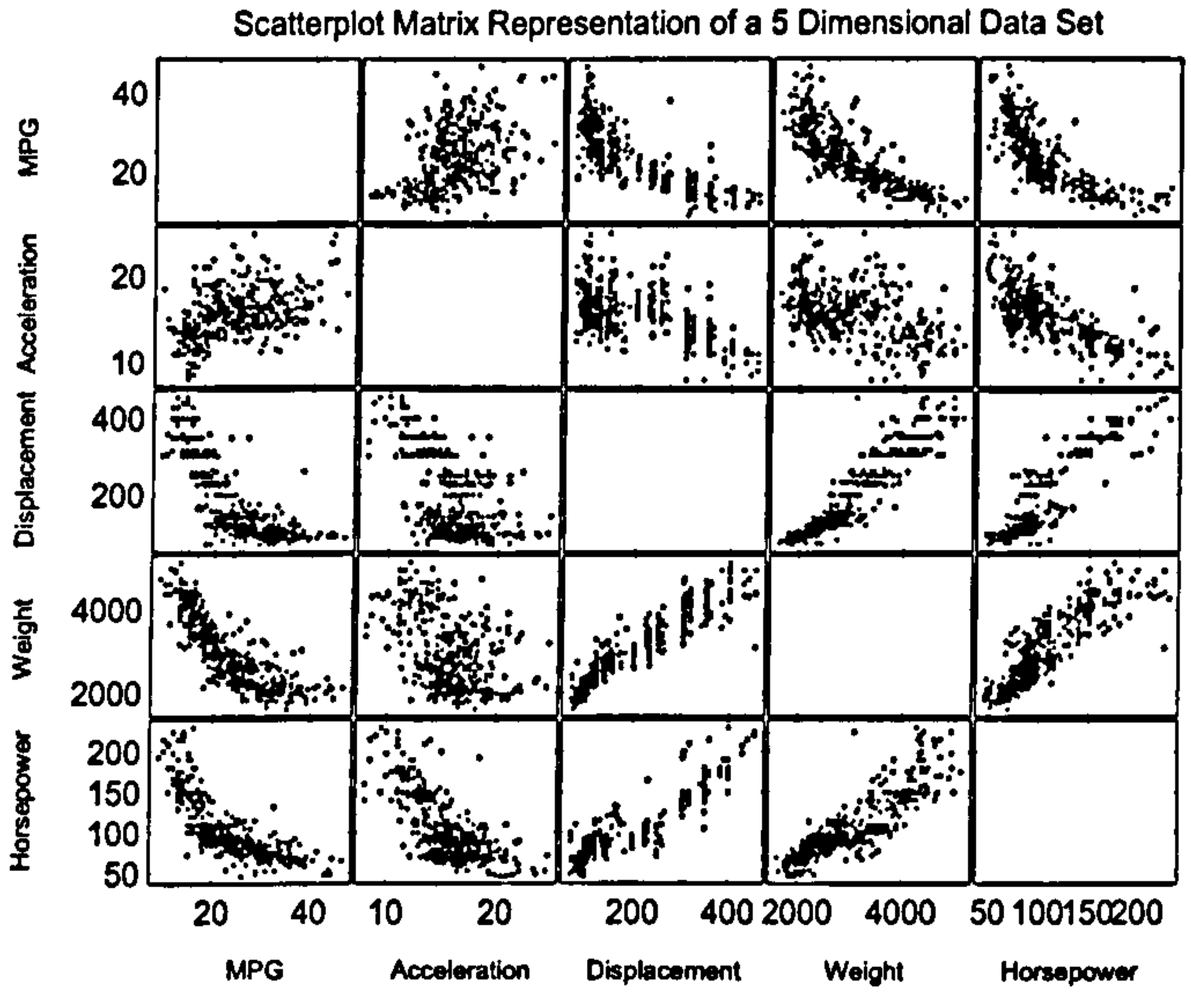


Figure 6.2: A Scatter Plot Matrix Representation of a Five Dimensional Data Set

Each star consists of equally spaced radii extending from the centre of a circle, with each radii representing one of the dimensions of the data (see Figure 6.3). The major weakness in this kind of multi-dimensional visualisation technique is that they are not useful for large data-sets, since each point in the data-set requires a star to be constructed. This gives them a representational complexity of $O(m)$, where m is the number of points in the data-set. Their interpretation is also subjective, since there is no quantitative information about the data displayed.

Parallel Coordinate Plots (Inselberg 1985) are also commonly used for visualising high dimensional data. They allow the visualisation of high dimensional data in a simple two dimensional representation. Instead of having the axes orthogonal to each other, as in Cartesian geometry, the axes are placed in parallel. Thus a point in n dimensional space will be represented as a line that bisects n parallel axes (see Figure 6.4). Figure 6.5 illustrates the mapping between data points in the Cartesian system and the corresponding data using a parallel coordinates representation, where points A and B in the Cartesian system are represented by the bold lines in the parallel coordinate plot.

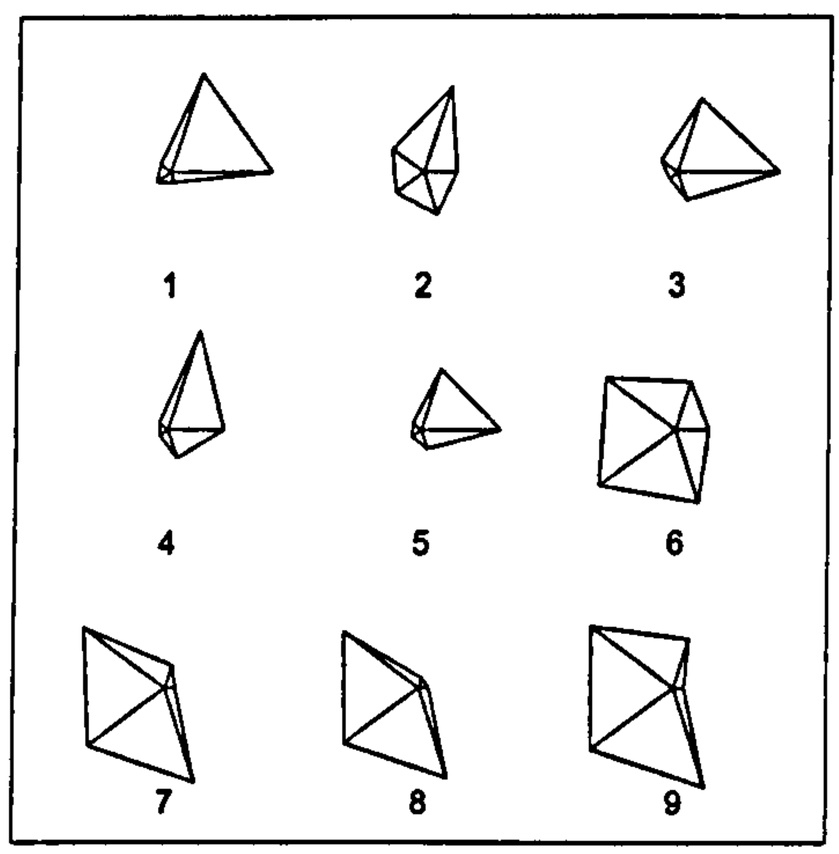


Figure 6.3: A Star Plot Representation of a Five Dimensional Data Set

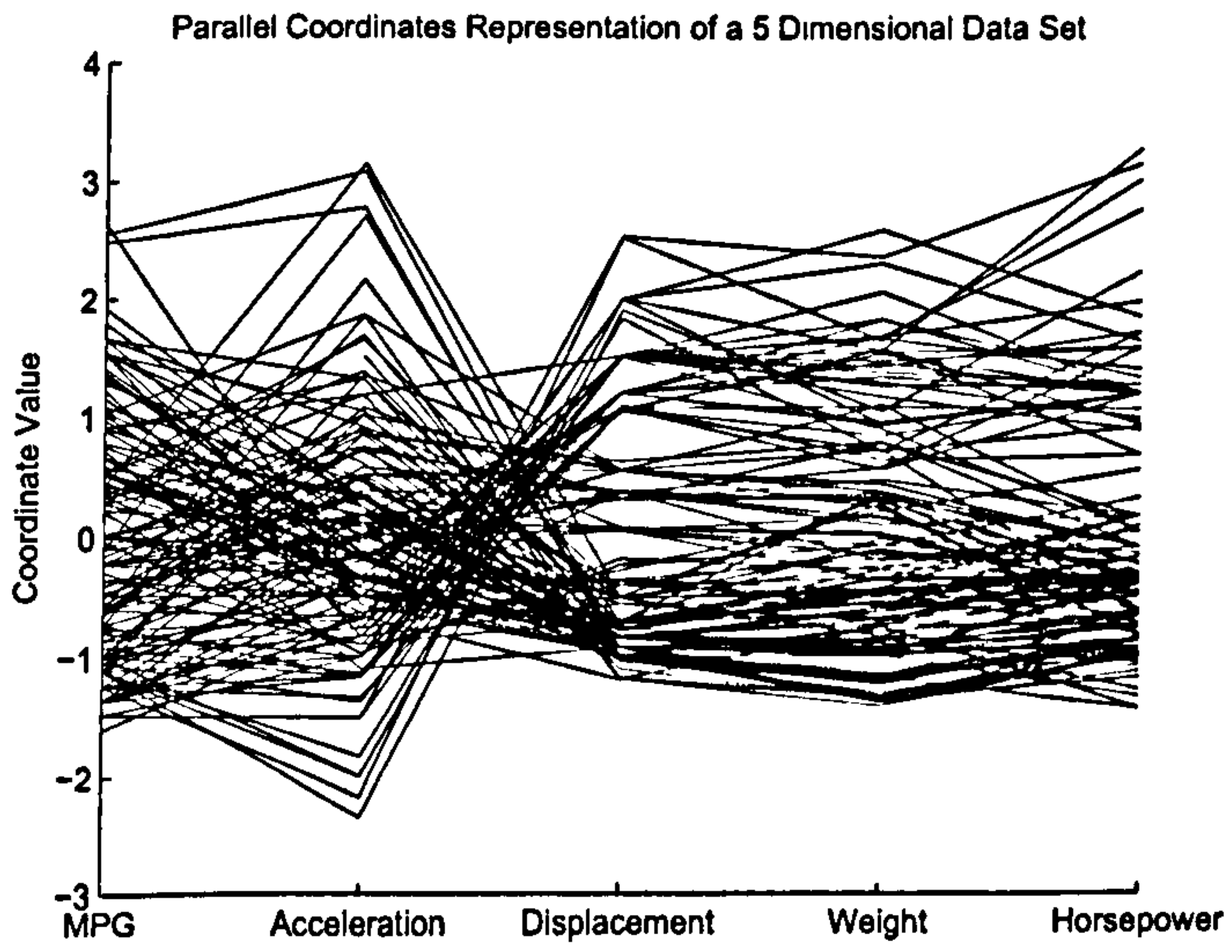


Figure 6.4: A Parallel Coordinate Plot Representing a Five Dimensional Data Set

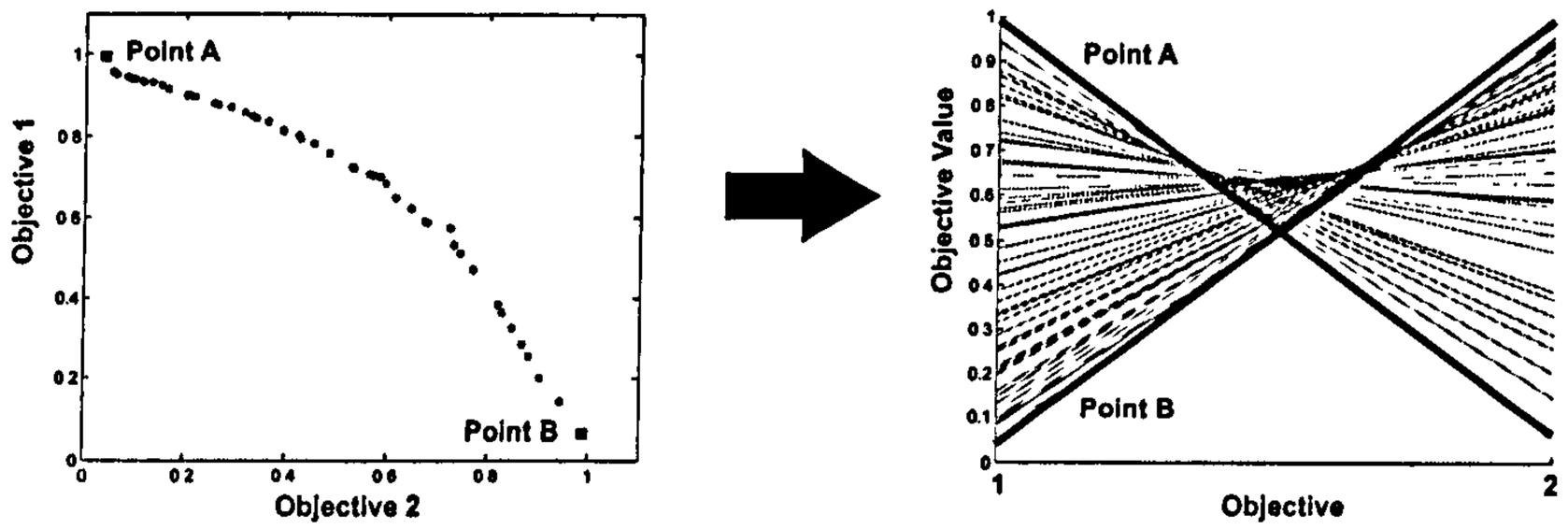


Figure 6.5: Mapping Between the Cartesian System and the Corresponding Parallel Coordinate Plot

Parallel Coordinate plots have a lower representational complexity than both scatter plot matrices and star plots. Their representational complexity is $O(n)$ (where n refers to the dimensionality of the data), as each line bisecting the axes of the plot represents a single point in high dimensional space. Parallel coordinates also lose no data in the representation process; this in turn ensures that there is a unique representation for each unique set of data. The main weaknesses of this method are that it requires multiple re-orderings to see different trade-offs (since trade-offs can only be seen between objective that are next to each other) and it can be difficult to distinguish individual points if many data points are represented.

Heatmaps are a high-dimensional visualisation technique that is commonly used in biology for visualising gene expression data (Cook *et al.* 2007). Heatmaps use colour intensity to represent the magnitude of each data point in each dimension, and so, like parallel coordinates, they have a representational complexity of $O(n)$. However, they typically require the use of a clustering algorithm (such as k-means clustering) to order the data in such a way that it can be meaningfully interpreted. For example, the heatmap in Figure 6.6 is ordered with regards to the first objective. Using this ordering it can be seen that objectives 4 and 5 are fairly well correlated with this first objective, whilst objectives 2 and 3 conflict with it. However, without any ordering this plot would be very difficult

to interpret. The main weakness with this visualisation method is how much its interpretability relies on the reordering strategy used. It is also often difficult to understand the distribution of values because the resulting heatmaps are not geometrically interpretable (Cook *et al.* 2007), and, like parallel coordinates, multiple re-orderings are required to see different relationships between the data.

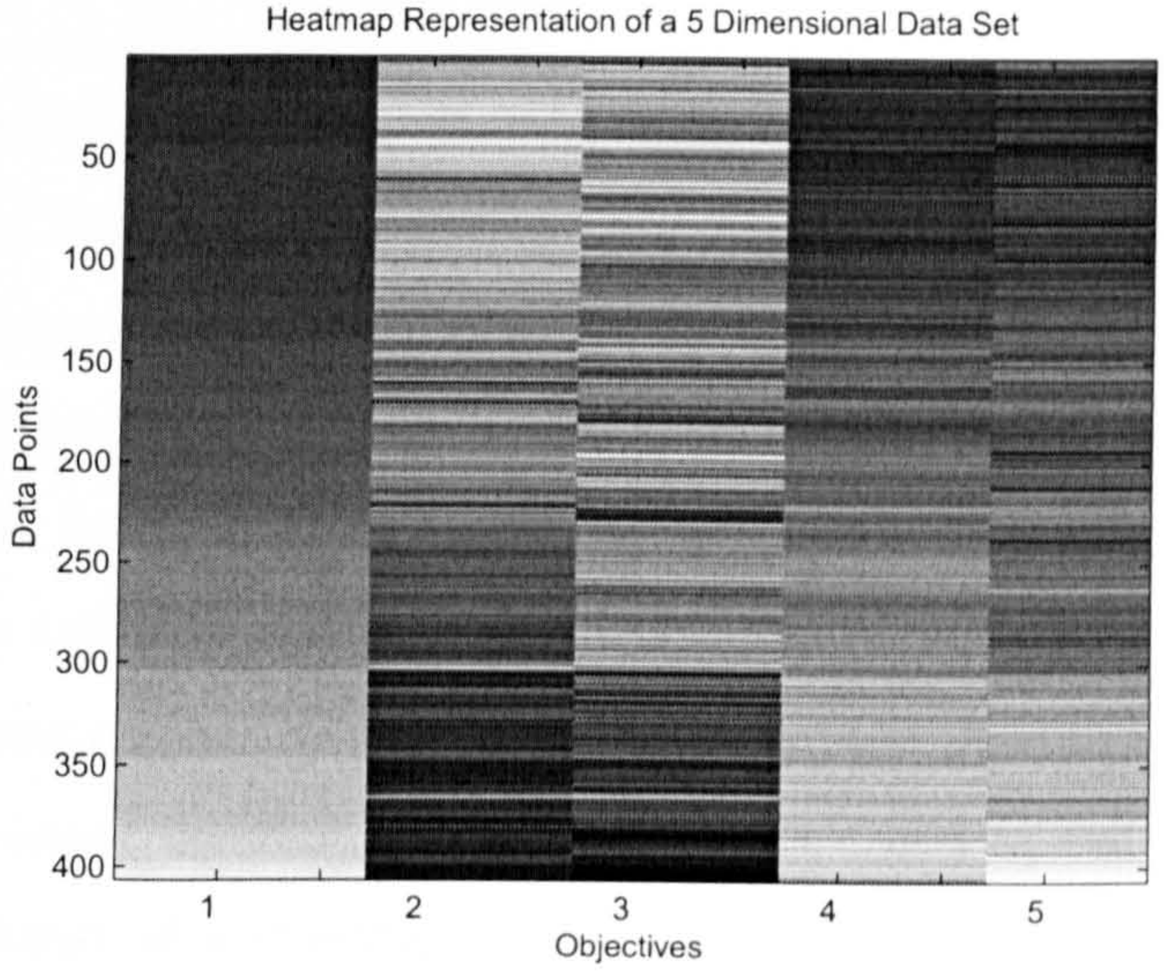


Figure 6.6: A Heatmap Representation of a Five Dimensional Data Set

Parallel Coordinate plots were chosen to perform the visualisation of the data in the proposed computational steering system due to their ease of interpretation and the ability to display all the appropriate data on a screen of limited size, for example on a PDA. To overcome the potential problem of having difficulty distinguishing individual points when the display is cluttered, only those candidate solutions that fit the DM's modified definition of Pareto optimality (see Section 6.3) will be displayed. This will prevent the plot from becoming cluttered without removing any of the useful data, i.e. the candidate solutions in the region of interest of the decision maker.

6.4.3 A PDA Implementation

In the application domain of Engineering Design it would be especially useful for an engineer working on a many-objective optimisation problem to be able to check on the progress of the algorithm from the field. An ideal client for this computational steering system would provide low-cost, portable access to the system.

The implementation of this steering client can be effectively realised by using a PDA enabled with a wireless connection. The low cost involved in the use of a PDA-based client would allow a wide uptake of this steering system by engineers in the field, whilst the portability of a PDA-based client would enable the engineer to check on the progress of the optimisation routine and alter parameters from anywhere with access to the internet. This would be especially useful in the case of a long running optimisation process that may take days to complete.

This PDA-based steering system was implemented in two parts. The first part was the design of a steering-enabled multi-objective evolutionary algorithm web service. This service provides the ability to adjust the parameters of the optimisation routine as well as allowing the execution of the multi-objective evolutionary algorithm for a given number of generations. The second part of the implementation was the development of a PDA-based client to interface with the steering-enabled multi-objective evolutionary algorithm web service. Due to issues with scarcity of memory and computational power, this client has to be lightweight whilst still providing the desired functionality.

The PDA-based client connects wirelessly to the steering-enabled multi-objective evolutionary algorithm web service. The decision maker can then choose whether to initiate a new optimisation routine or continue a previous one. The DM can then adjust the parameters of the optimisation process and run the multi-objective evolutionary algorithm for a given number of generations. The results are displayed in the PDA client as a parallel co-ordinate plot of the objectives (see

Figure 6.7), and this information can then be used to improve the optimisation process.

Figure 6.7: A PDA Based Implementation of the Steering Client for a Multi-Objective Evolutionary Algorithm

6.5 Application of a Computational Steering System for Multi-Objective Optimisation

A many-objective aircraft controller design problem from literature (Tabak *et al.* 1979) was chosen to illustrate the process of computational steering of a multi-objective evolutionary algorithm. This problem involves the design of controller gains to obtain rapid and precise roll response to aileron inputs. There are 8 objectives:

1. Control Effort
2. Bank Angle at 2.8 seconds
3. Side Slip Deviation
4. Spiral Root
5. Roll Damping Root
6. Dutch Roll Damping Ratio
7. Dutch Roll Damping Frequency
8. Bank Angle at 1 second

Initially a multi-objective evolutionary algorithm with common values for the parameters was used to attempt to solve this problem. This algorithm was run for 100 generations with no preference articulation (see Figure 6.8 and Figure 6.9). The mutation rate was then reduced, and the bounds on some of the decision variables (those that can be seen, on inspection of the results, to contribute to an unsatisfactorily large value for Objective 3) tightened. The algorithm was then re-run (again for 100 generations - see Figure 6.10).

Altering the parameters of the multi-objective evolutionary algorithm improved the results produced by the optimiser, but the solutions are still not satisfactory. *A priori* preference articulation was then incorporated into the algorithm (see Section 6.3), and the MOEA with *a priori* preference articulation was run for 100 generations. These results are shown in Figure 6.11.

As can be seen from Figure 6.11, the incorporation of *a priori* preference articulation helps the algorithm to achieve better results (from the DM's point of view). This is because the initial preferences given to the algorithm impose a strict ROI on which to focus the search. However, these solutions can be further improved by computational steering of the optimisation process. The

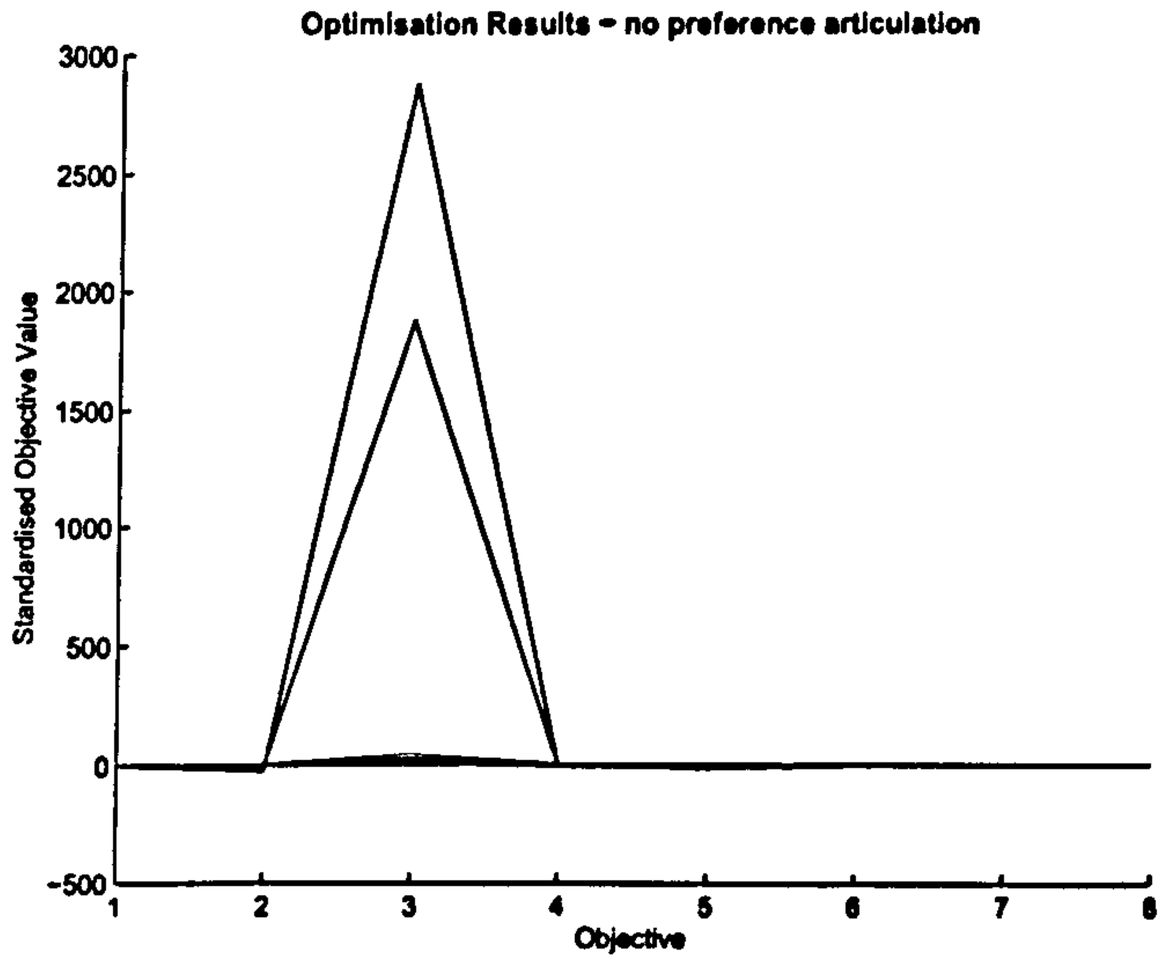


Figure 6.8: Results of the Initial Execution of the Multi-Objective Evolutionary Algorithm

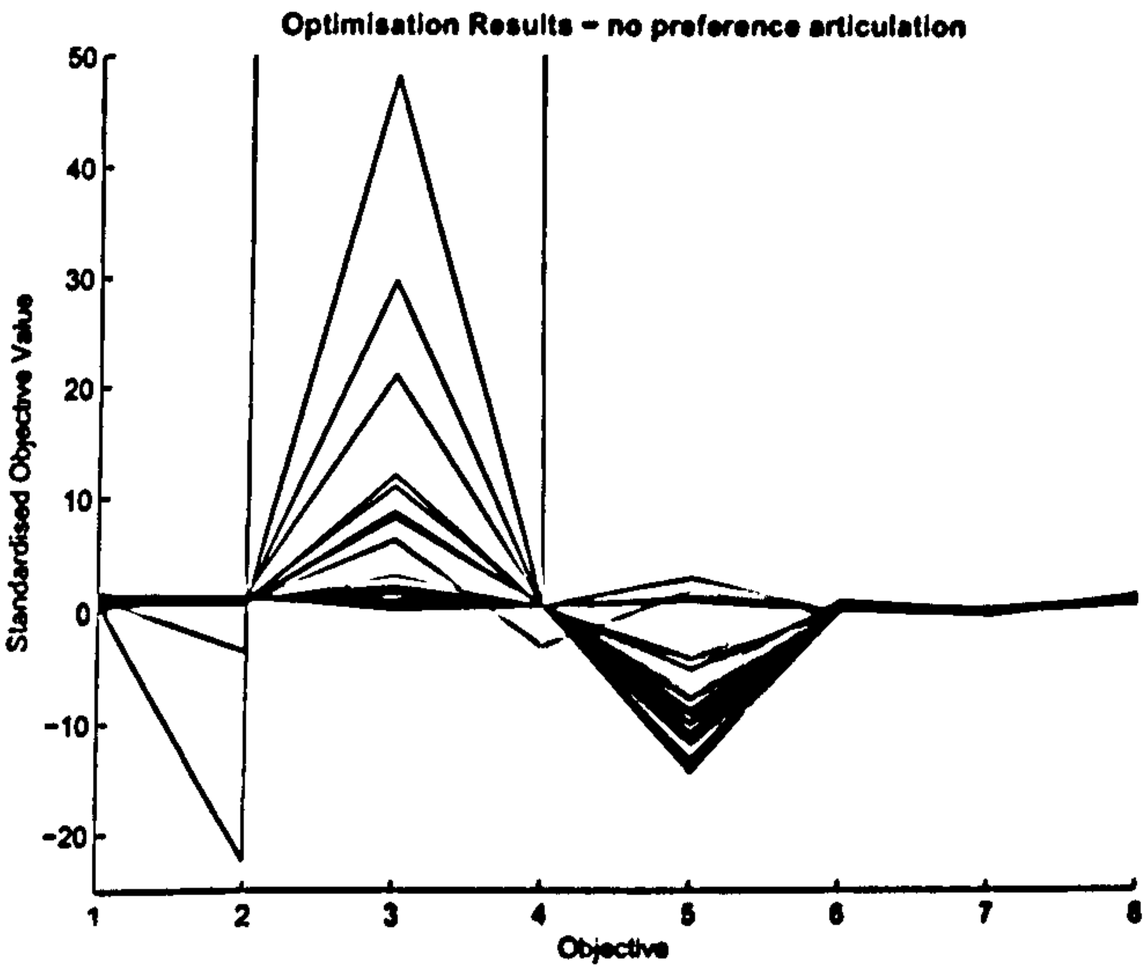


Figure 6.9: A Close Up View of the Results of the Initial Execution of the Multi-Objective Evolutionary Algorithm

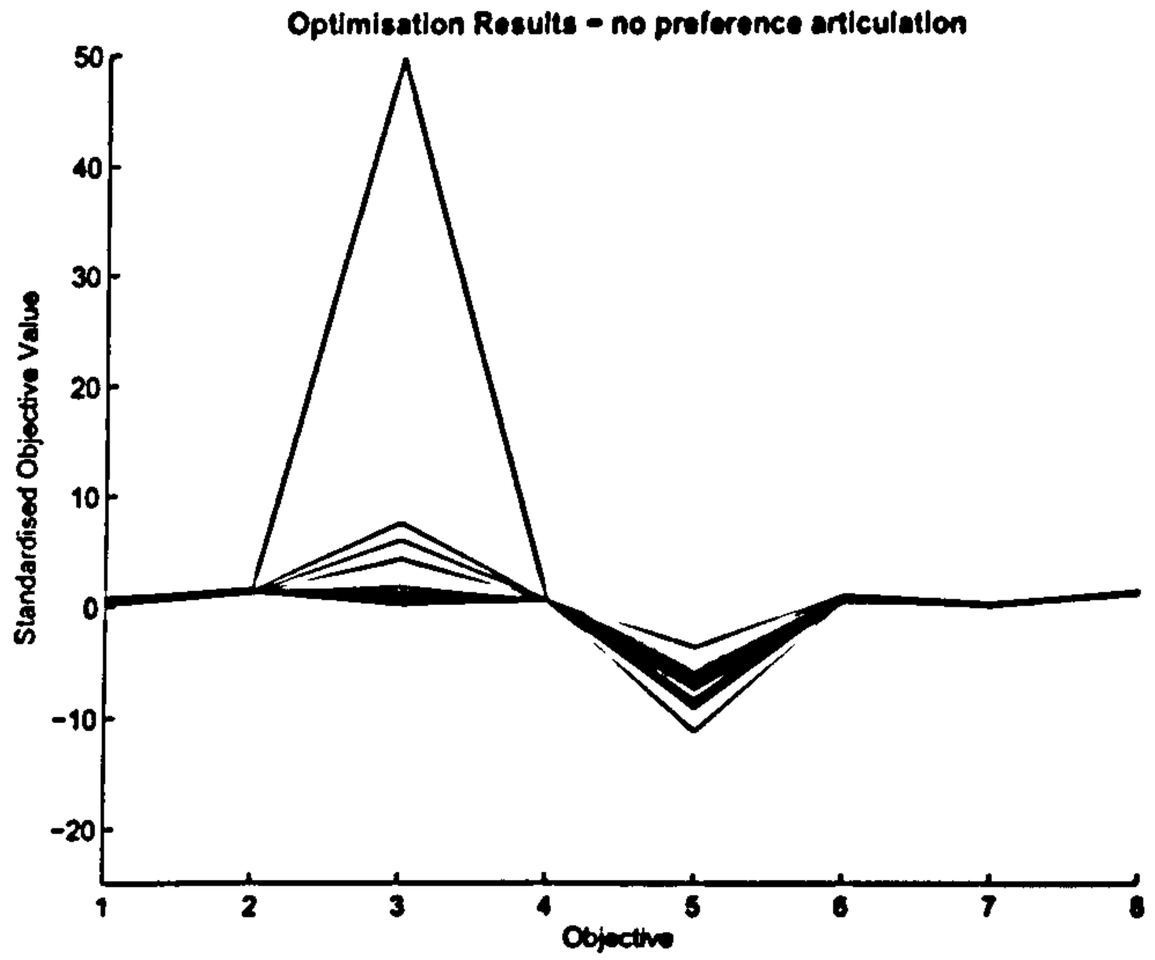


Figure 6.10: Results of the Re-execution of the Multi-Objective Evolutionary Algorithm

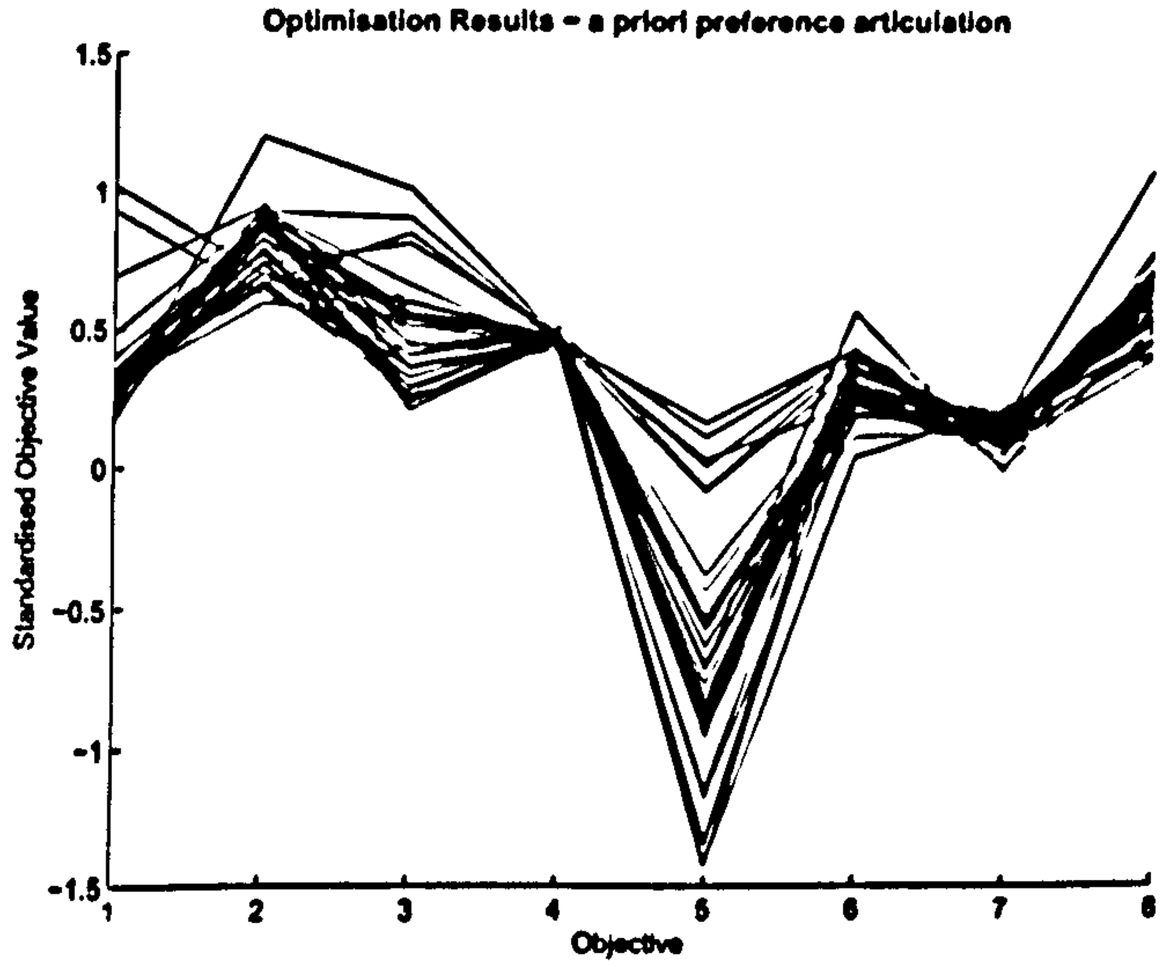


Figure 6.11: Results of the Multi-Objective Evolutionary Algorithm with *a priori* Preference Articulation

computational steering of the MOEA was carried out using the steering-enabled web service described in Section 6.4.3, however a desktop implementation of the steering client was used so as to produce the figures in this Chapter. The results were confirmed using the PDA client.

Figure 6.12 is a parallel coordinates representation of the 8 objectives after having executed the steering-enabled multi-objective evolutionary algorithm for 20 generations. The dashed line represents the initial goal values for the algorithm and therefore the solutions shown on the plot are those that satisfy the initial design specifications, i.e. are within the ROI defined by the DM.

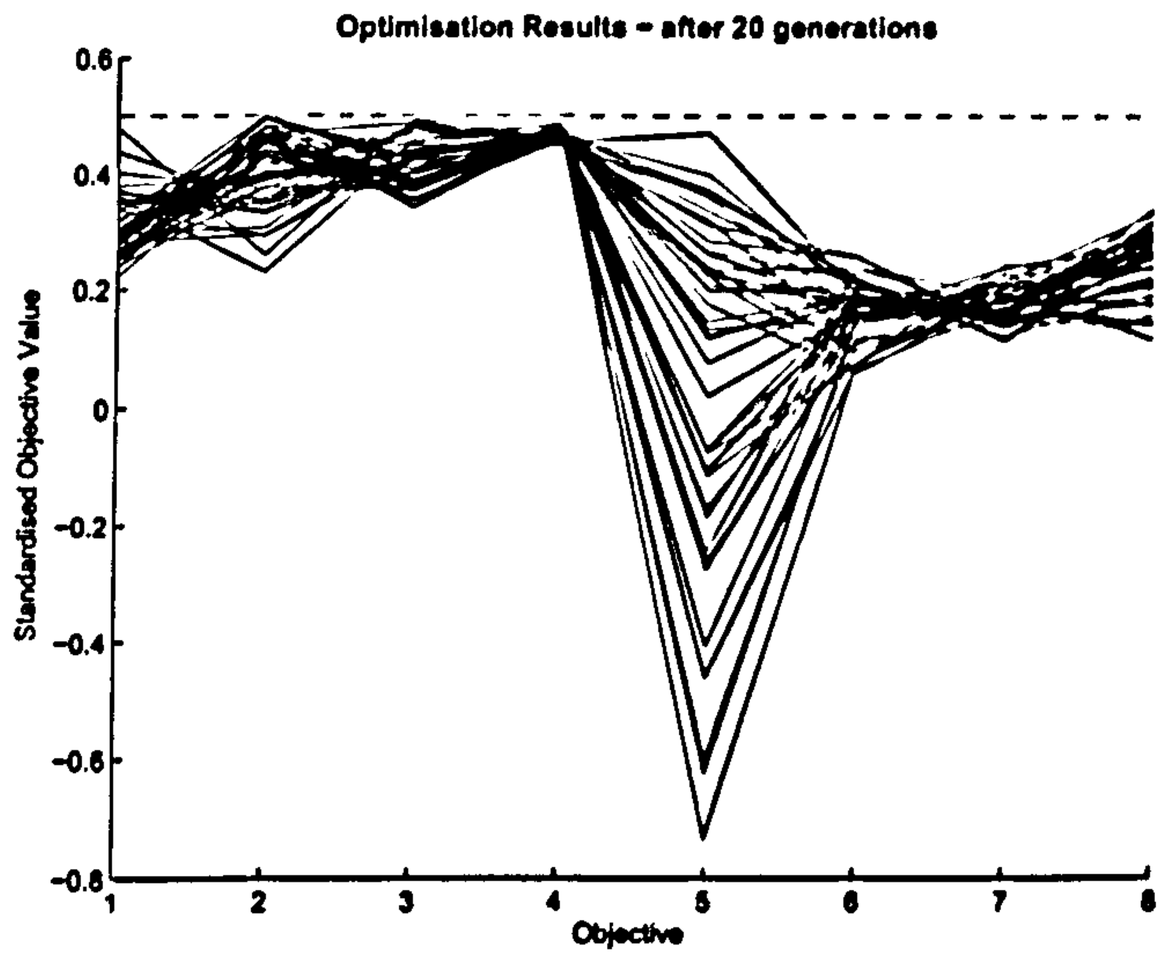


Figure 6.12: Results of the Multi-Objective Evolutionary Algorithm after 20 Generations using the Computational Steering Client

The DM knows from domain knowledge that it is important to keep the control effort (Objective 1) small. This is because high gains can cause sensitivity to sensor noise and may lead to saturation of control actuator response. The decision maker therefore constrains this objective to be at least as good as it is at the moment. This plot also shows that the DM can tighten the goals on objectives 5, 6, 7, and 8. The multi-objective evolutionary algorithm is then run for another 10 generations (see Figure 6.13).

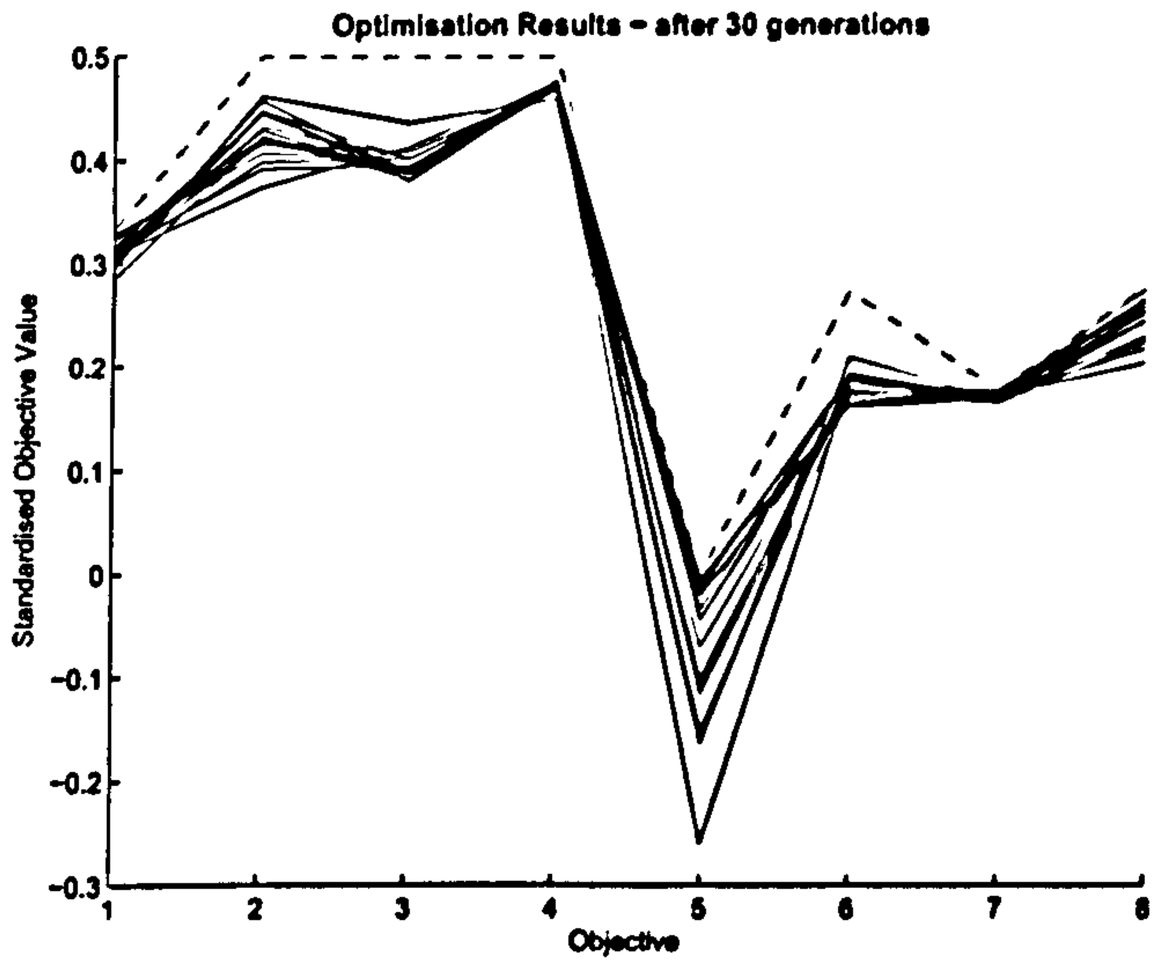


Figure 6.13: Results of the Multi-Objective Evolutionary Algorithm after 30 Generations using the Computational Steering Client

A decision is now made to isolate the best solution (see Figure 6.14) with respect to the Roll Damping Root (Objective 5). This objective is of primary importance for aileron response. Ideally this objective will provide a fast well damped response to the control signals. The mutation rate of the multi-objective evolutionary algorithm is then turned down so as to generate multiple solutions that are close to this (see Figure 6.15).

This provides the decision maker with multiple solutions to choose from. The DM then picks the best solution with respect to the Bank Angle at 2.8 seconds (Objective 2) as this ensures that the speed and steadiness of the basic roll response does not drop off with time (see Figure 6.16).

Figure 6.17 shows the 'best' solution produced by each of the runs of the multi-objective evolutionary algorithm[1]. The solutions plotted for the first three MOEA runs are those chosen by the decision maker *a posteriori* (see Section 6.3) according to the preferences of the DM. The fourth solution plotted on the graph

[1]The results presented here are from a single run of each of the MOEAs, however similar results were obtained from repeated experiments.

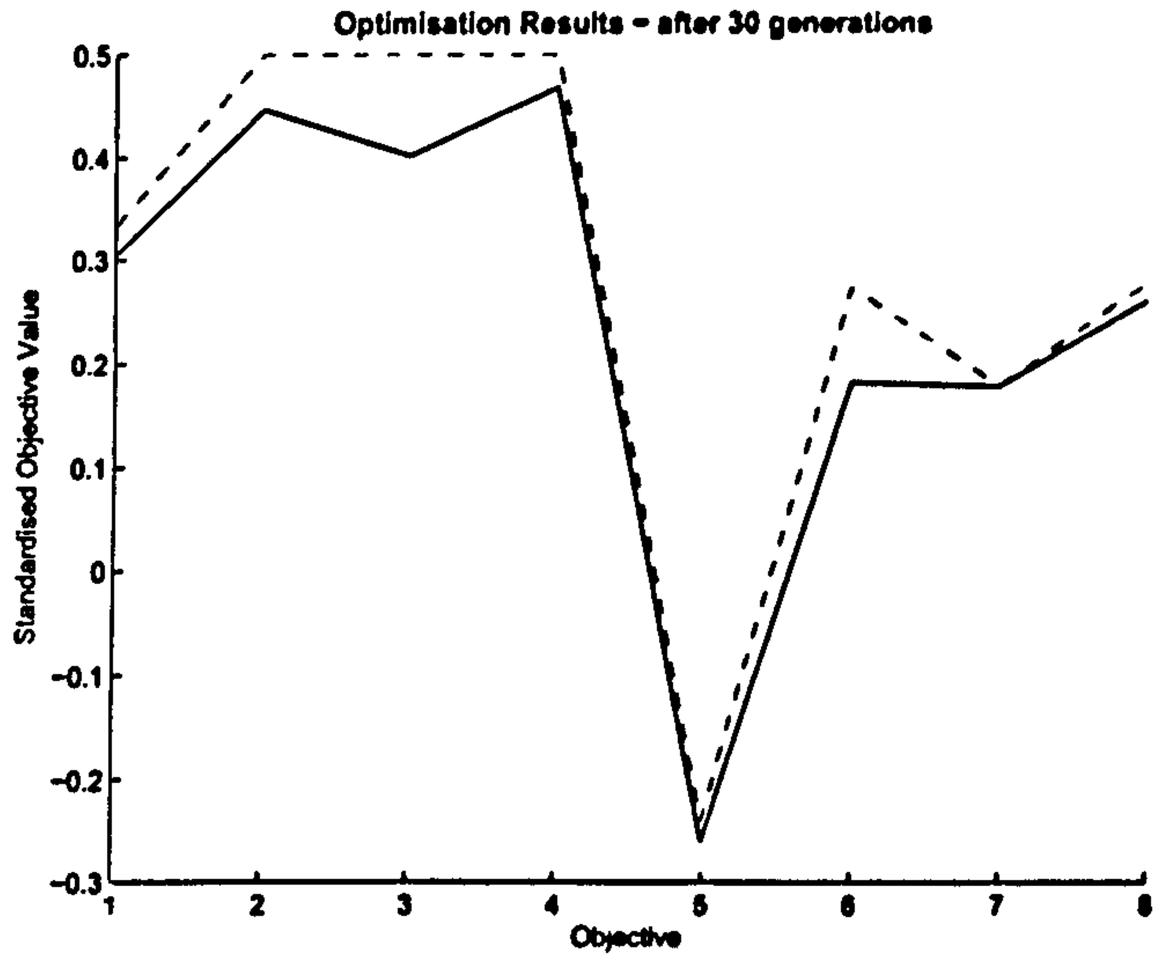


Figure 6.14: Isolating a Solution from the Steered Multi-Objective Evolutionary Algorithm

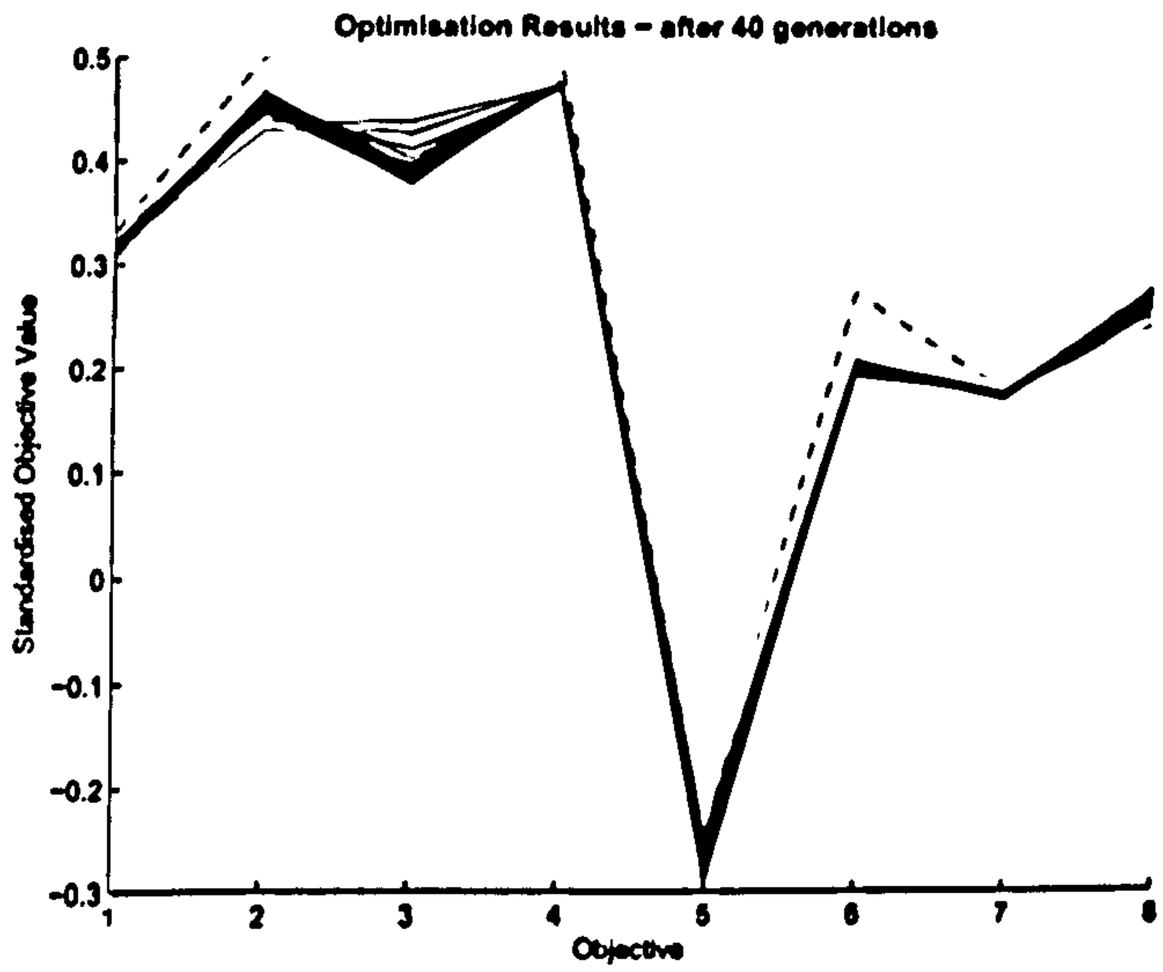


Figure 6.15: Results of the Multi-Objective Evolutionary Algorithm after 40 Generations using the Computational Steering Client

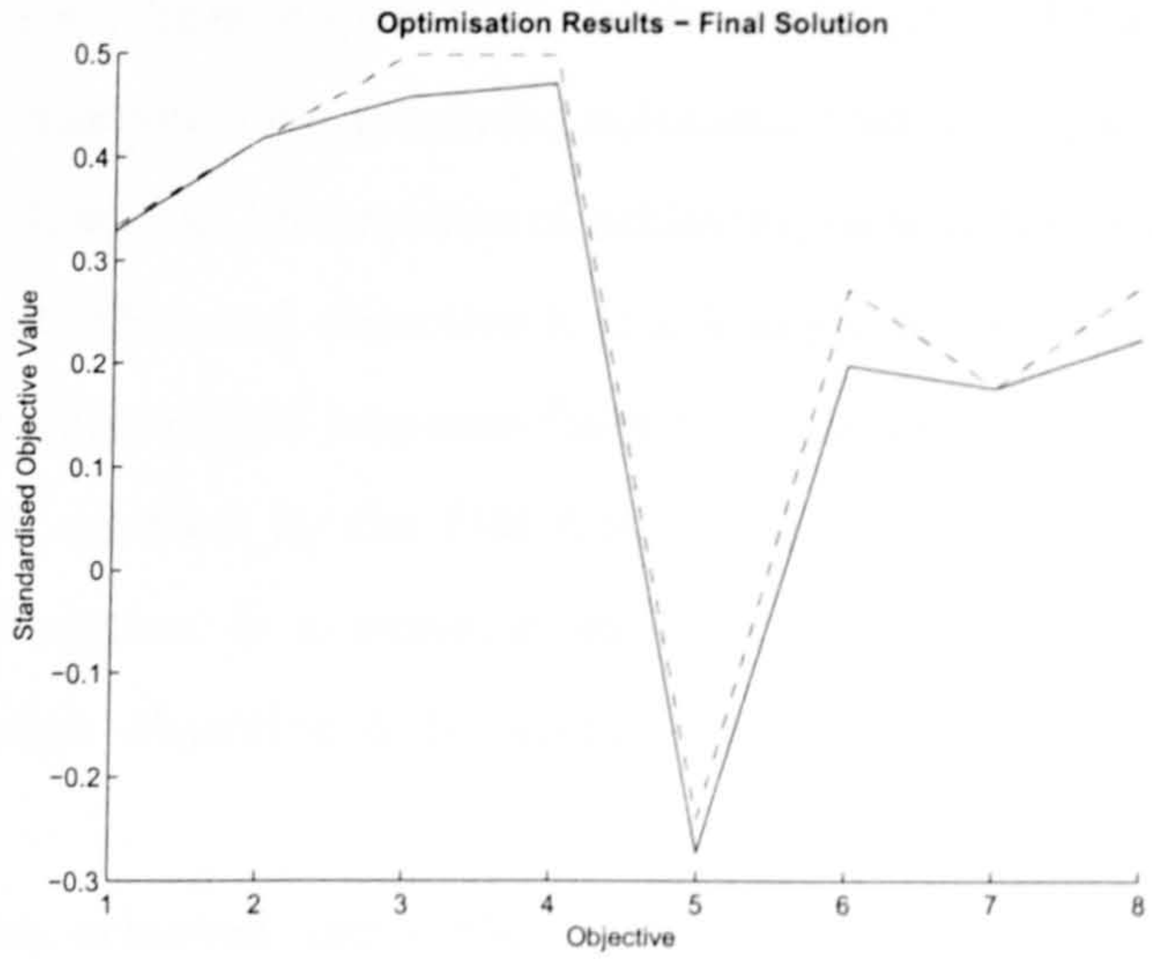


Figure 6.16: The Final Solution Chosen by Decision Maker from the Steered Multi-Objective Evolutionary Algorithm

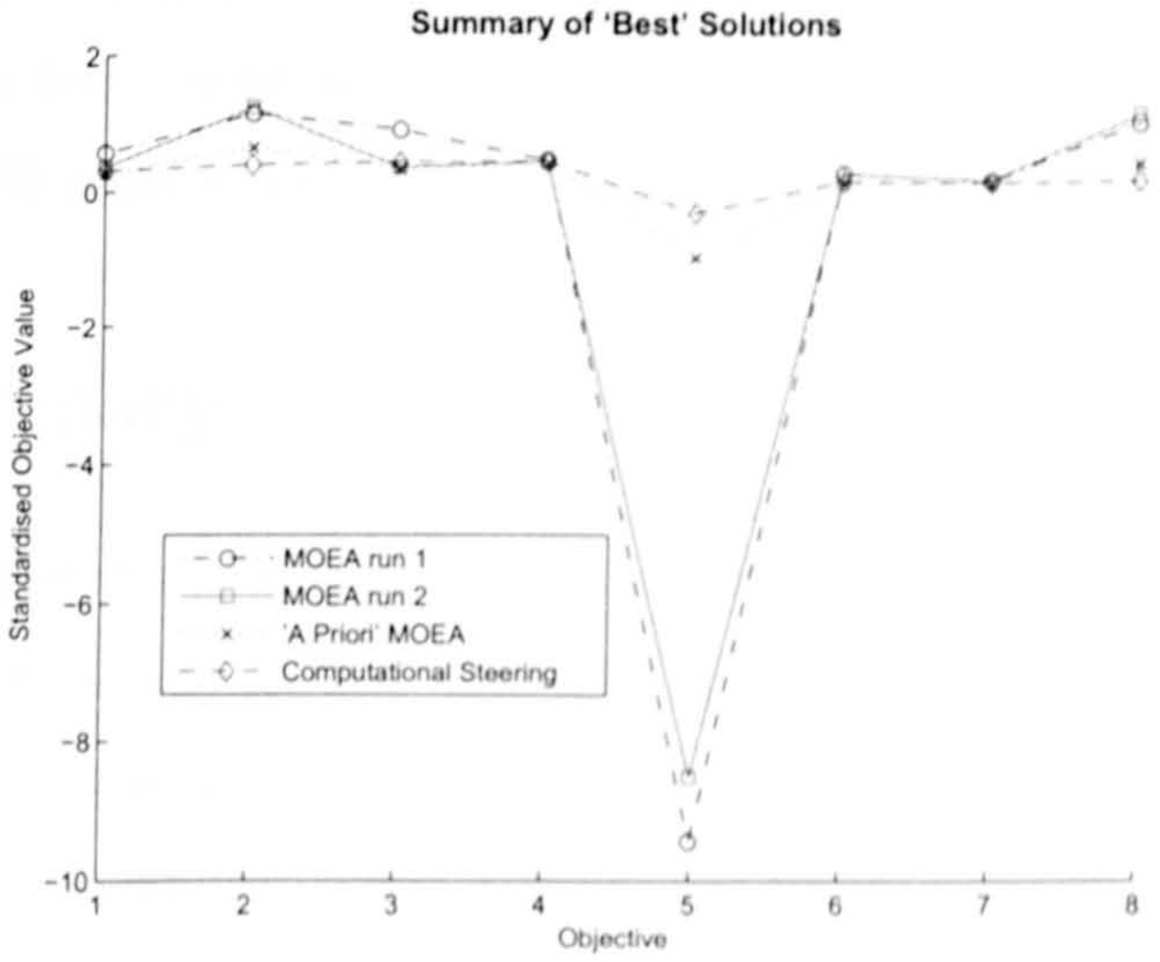


Figure 6.17: The Solutions Chosen by the Decision Maker from the Pareto Optimal Sets Produced by each of the MOEAs

is the final solution achieved by the proposed computational steering system (see Figure 6.16).

As can be seen from Figure 6.17, both of the runs of the MOEA with no preference articulation have produced solutions that minimise objective 5 very well. However, this is at the expense of achieving satisfactory values for objective 2 (bank angle at 2.8s) and objective 8 (bank angle at 1s). These objectives are important to ensure a good response from the controller.

The solution selected by the DM from the run of the MOEA with *a priori* preference articulation is a superior solution (from the DMs point of view) because, although objective 5 is larger, objectives 2 and 8 are much more satisfactory.

The solution achieved using the proposed computational steering system provides the best response of all the solutions. It is the most successful solution in minimising objectives 2 and 8, whilst still achieving good values for the other objectives. The proposed computational steering system also produced this solution in fewer generations than the other MOEA runs (40 generations, compared to 100 generations for each of the other algorithms).

6.6 Summary

The execution process of a evolutionary algorithm typically involves some trial-and-error. This is due to the difficulty in setting the initial parameters of the algorithm - especially when little is known about the problem domain. This problem is magnified when applied to many-objective optimisation, as care is needed to ensure that the final population of candidate solutions is representative of the trade-off surface. This Chapter proposes a computational steering system that allows the engineer to interact with the optimisation routine during execution. This interaction can be as simple as monitoring the values of some parameters during the execution process, or could involve altering those parameters to influence the quality of the solutions produced by the optimisation process.

Section 6.1 has introduced some of the key concepts behind the idea of computational steering, as well as outlining some recent applications where computational steering has enabled engineers and scientists to interact with complex long running simulations. The importance of visualisation in the steering process was emphasized, and so, in Section 6.4.2, a survey of commonly used high-dimensional visualisation techniques was performed. An important issue in the application of computational steering to evolutionary multi-objective optimisation is the way that a decision maker expresses their preferences. Section 6.3 briefly summarises the different types of preference articulation (explained in more detail in Section 2.4.5), before presenting the results of a recent survey of interactive preference articulation methods (Adra *et al.* 2007).

Section 6.5 shows that significant improvements in both the execution speed of the MOEA (i.e. the number of generations needed to find good solutions) and the quality of the solutions produced by the MOEA can be achieved by using computational steering to guide the optimisation routine. By using both progressive preference articulation (Fonseca and Fleming 1998) and steering of parameters, such as population size and mutation rate, the need for repeated execution of the multi-objective evolutionary algorithm can be avoided. This improves the efficiency of the optimisation process.

The results produced by computational steering of the optimisation routine are an improvement over those achieved without using steering. This is due, in part, to the ability to alter the goals progressively. However, being able to alter the mutation rate and other parameters is an important factor. For instance, in order to produce multiple solutions in close proximity to each other (see Figure 6.15), the DM was able to reduce the mutation rate which decreased the amount of variation produced in the next generation of candidate solutions.

The ability to control the population size of the algorithm also proved useful, as a large population size can be used initially, so as to cover a large area of the search space, and then the population size can be reduced to allow for quicker

execution of the algorithm.

Although altering the bounds on the decision variables was unnecessary in the case of the example in Section 6.5, providing the ability to do so is potentially useful. By changing the upper and lower bounds the decision maker is able to guide the search in decision space, as well as guiding the search in objective space using preference articulation. By loosening the bounds on certain decision variables, the DM can expand the search to include previously ignored areas, whereas by tightening the bounds the DM can constrain the area of decision space searched by the optimiser.

This computational steering process for evolutionary optimisation may also allow the user to gain some additional insight into the optimisation process. As has been noted in the introduction to this Chapter, the evolutionary computation community possesses little more than 'rules-of-thumb' when it comes to setting the initial parameters of an evolutionary algorithm (Bullock *et al.* 2002). By using computational steering of the evolutionary optimisation process, it may be possible to further understand the effects and interactions of the different parameters in the algorithm.

Chapter 7

Conclusions

Modelling, optimisation and decision support are key areas that need to be effectively addressed in the engineering design domain. Soft computing techniques such as neural networks and evolutionary computation can often provide the engineer with a powerful set of tools for tackling these issues, frequently outperforming conventional methods. Evolutionary optimisation especially has been shown to effectively assist decision making in engineering design, with multi-objective evolutionary algorithms allowing an engineer to address many design criteria simultaneously.

The main drawback with these techniques is that they tend to be computationally expensive. Evolutionary optimisation methods in particular may require many thousand candidate solution evaluations to arrive at a satisfactory final solution and, with the evaluation of candidate solutions in many real-world problems being performed by complex computer simulation, this optimisation process can take many hours or days to complete. It is no surprise, therefore, that recently there has been growing interest in accelerating this optimisation process.

7.1 Grid Computing in Science and Engineering

The recent paradigm of Grid computing offers one potential solution to the computationally expensive nature of these techniques. Grid computing aims to provide "a seamless, integrated computational and collaborative environment" (Baker *et al.* 2002) that is ideal for addressing engineering design problems. The vision motivating Grid computing is that of computing as a utility, where engineers and scientists have transparent access to large amounts of compute resources 'on demand'. Chapter 4 has sought to bring this vision closer to reality by introducing a framework for the design of application-centric metaschedulers to enable transparent access (from an application user's point of view) to Grid resources located across multiple, geographically distributed institutions. This metascheduling framework is independent of both the underlying architecture of the Grid resources and the local resource management systems used.

Many applications in science and engineering - such as evolutionary optimisation methods - require the iterative processing of multiple independent tasks. A key contribution of Chapter 4 was therefore the development of a novel optimal workload allocation algorithm for minimising the response time of batches of jobs. Much of the previous analytical work in the literature has concentrated on individual arrivals of jobs, and this was shown in Section 4.5.4 to be suboptimal for the arrival of multiple jobs simultaneously.

This optimal workload allocation algorithm was coupled with a novel hybrid approach to job scheduling that combines the low computational cost of static scheduling policies with the ability to adapt to changes in the processing capacity of available Grid resources. This novel approach to job scheduling overcomes the need to explicitly query the Grid resources for their status (a process that can be both complex and computationally expensive), and instead uses information from previously completed tasks to estimate the current computational capacity of the resources in the Grid.

7.2 Parallel Evolutionary Computation on the Grid

Chapter 5 has shown the potential for addressing engineering design problems using the power of computational Grids to accelerate evolutionary computation. A key issue in the design of the proposed evolutionary optimisation framework was the model of parallelism to use for the evolutionary algorithm. A major contribution of Chapter 5 was in performing a rigorous performance comparison between the two main parallelisation strategies on a variety of single-objective and multi-objective test functions from the literature, and showing that many of the results showing the superior performance of island model EAs may be misleading when they are applied to real-world problems.

The usefulness of the Grid-enabled evolutionary optimisation framework developed in Chapter 5 in solving engineering design problems was proven by the optimisation of two computationally expensive real-world problems. The results presented in Chapter 5 show that, for problems with computationally expensive objective function evaluations (such as those performed by computer simulation), substantial speed ups can be achieved. It can also be seen that the potential speed ups achievable using the proposed framework increase with the complexity of the objective function evaluations.

7.3 Computational Steering of Evolutionary Multi-Objective Optimisation

Another potential solution to the computationally expensive nature of using evolutionary optimisation techniques for addressing problems in engineering design is the use of computational steering to allow a decision maker to guide the optimisation process. The difficulty in setting the initial parameters in

an evolutionary algorithm - especially in problems with many objectives and where little is known about the search landscape - means that it is common practice for the algorithm to be run until a certain termination condition is met and then the solution set analysed. If the results are not satisfactory then the parameters of the algorithm are then altered and the algorithm is re-run. This process of repeated execution of the MOEA clearly leads to an inefficient use of computational resources, and possibly also to inferior solutions.

The computational steering system proposed in Chapter 6 aims to overcome this problem by providing a decision maker with a way of influencing the search and altering the parameters of the algorithm on-line. The computational steering system uses progressive preference articulation methods to allow a DM to focus the algorithm on regions of the search space that are of interest, thus reducing the computational effort spent searching areas that are unlikely to produce solutions that are desirable to the decision maker. This system allows a decision maker to refine their preferences over the course of the search, potentially starting with very little knowledge about the achievable solutions for the problem. The ability to alter the other parameters of the algorithm is also an important factor in the effectiveness of the proposed system. For example, by reducing the mutation rate in the MOEA the convergence of the population can be speeded-up.

7.4 Future Challenges

7.4.1 Running Applications on the Grid

There is still a long way to go until the vision of Grid computing as a utility becomes a reality. Although the Grid computing paradigm is slowly becoming more widespread, it is still some way from providing transparent access to computational resources located across the globe. Providing this transparent access to compute resources is one of the key challenges facing the Grid

computing community. To provide this will require advances in Grid middleware, specifically in the ability to discover and use resources that lie outside institutional boundaries.

Another major challenge facing the Grid computing community is managing security in an 'on demand' network of compute resources. A key issue in Grid security is the implementation of *restricted delegation* policies that will allow user applications to run with the minimum set of privileges needed to accomplish their task (Ahsant *et al.* 2006). This is important because allowing a user application or service full privileges increases the risk of abuse or attack, whilst delegating too few privileges means the application may not be able to complete its task.

As was shown in Chapter 5, plenty of scope for improvement also exists in guaranteeing a minimum level of service for applications on the Grid. Whilst advances are being made in this area with recent research into Service Level Agreements (SLAs) (Yarmolenko and Sakellariou 2007), there is still much work to be done. Without a guarantee of a minimum level of service it is unlikely that there will be wide spread adoption of the Grid computing paradigm outside of academia.

7.4.2 Evolutionary Computation

Although the computational steering framework introduced in Chapter 6 achieved substantial improvements in both the time taken by the optimisation process and the quality of the final solution set produced by the optimiser, it is very DM intensive. To achieve good results using such a system it is important to get constant input from the decision maker, and thus some way of automating this process would be desirable. Todd and Sen (1999) have proposed the use of an artificial neural network to model the decision maker's preferences and thus guide the search; however, there is much further work to be done to automate this process effectively for many objectives.

Elitism has been identified as an important factor in the current generation of evolutionary multi-objective optimisers. However, it may have drawbacks when a decision maker wishes to progressively influence the search process (Zitzler *et al.* 2000). Recent research (Purshouse 2003, Shenfield, Fleming and Alkarouri 2007, Adra *et al.* 2007) has shown that the use of progressive preference articulation is an effective way of addressing problems with many objectives, and therefore further work is necessary to investigate the effect that using both elitism and progressive preference articulation has on the evolutionary multi-objective optimisation process.

Further investigation is also merited into the hybridisation of domain specific local search methods with evolutionary algorithms. This area is already attracting significant interest due to the potential increases in the efficiency of the EA that can be achieved. However, much of the work in this area is hindered by the *ad hoc* nature of the design process (Krasnogor and Smith 2005), and thus the development of a framework to assist in the design of these algorithms and the choice of which local search methods to incorporate would be a major asset.

It has been established in Chapter 5 that island model parallel evolutionary algorithms are ill-suited for execution in a Grid computing environment due to their sensitivity to the choice of extra parameters, especially the choice of the number of subpopulations to use. However, further investigation into their application to scalable test functions (such as those developed by Deb, Thiele, Laumanns and Zitzler (2002)) may be of interest, since there may be cases where island model EAs suit compute cluster architectures.

Appendix A

The M/M/1 Queue With Bulk Arrivals

Introduction

In this section we shall consider a single-server queueing system (illustrated in Figure A.1) consisting of a queue of infinite size and a single server capable of serving one request at a time. If the server is busy upon the arrival of a request, that request is put into the queue until the server becomes available. For a thorough introduction to queueing theory the interested reader is directed to Kleinrock (1975).

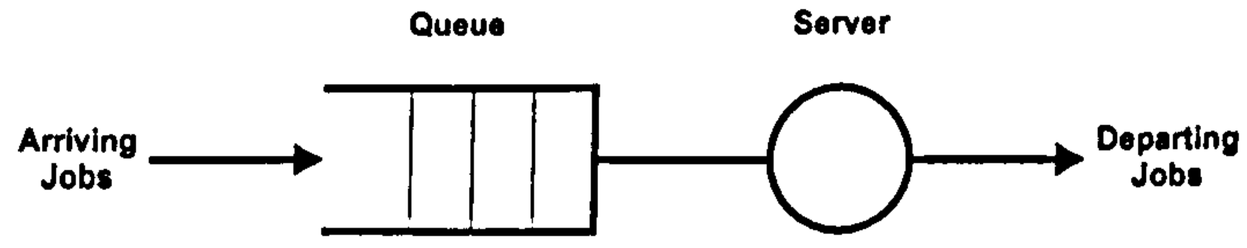


Figure A.1: A Single-Server Queueing System

The Transient Equations for the Queueing System

Arrivals occur in the system according to an exponential distribution with arrival rate λ, and service occurs according to an exponential distribution with service rate μ. Each arrival corresponds to a group of k requests. If $p_n(t)$ is equal to the probability of there being n requests in the system at time t, then the transition equations for the system can be written as Equations A.1.

$$\begin{aligned}
p_0(t+\Delta t) &= p_0(t)(1-\lambda\Delta t) + p_1(t)(1-\lambda\Delta t)(\mu\Delta t) \\
p_1(t+\Delta t) &= p_1(t)(1-\lambda\Delta t)(1-\mu\Delta t) + p_2(t)(1-\lambda\Delta t)(\mu\Delta t) \\
&\vdots \\
p_{k-1}(t+\Delta t) &= p_{k-1}(t)(1-\lambda\Delta t)(1-\mu\Delta t) + p_k(t)(1-\lambda\Delta t)(\mu\Delta t) \\
p_k(t+\Delta t) &= p_0(t)(\lambda\Delta t) + p_k(t)(1-\lambda\Delta t)(1-\mu\Delta t) + p_{k+1}(t)(1-\lambda\Delta t)(\mu\Delta t) \\
p_{k+1}(t+\Delta t) &= p_1(t)(\lambda\Delta t)(1-\mu\Delta t) + p_{k+1}(t)(1-\lambda\Delta t)(1-\mu\Delta t) + p_{k+2}(t)(1-\lambda\Delta t)(\mu\Delta t) \\
&\vdots \\
p_n(t+\Delta t) &= p_{n-k}(t)(\lambda\Delta t)(1-\mu\Delta t) + p_n(t)(1-\lambda\Delta t)(1-\mu\Delta t) + p_{n+1}(t)(1-\lambda\Delta t)(\mu\Delta t)
\end{aligned} \tag{A.1}$$

By rearranging the equations and letting $\Delta t \rightarrow 0$ we get the differential equations shown in A.2.

$$\begin{aligned}
\frac{\mathrm{d}p_0(t)}{\mathrm{d}t} &= -\lambda p_0(t) + \mu p_1(t) \\
\frac{\mathrm{d}p_1(t)}{\mathrm{d}t} &= -(\lambda+\mu)p_1(t) + \mu p_2(t) \\
&\vdots \\
\frac{\mathrm{d}p_{k-1}(t)}{\mathrm{d}t} &= -(\lambda+\mu)p_{k-1}(t) + \mu p_k(t) \\
\frac{\mathrm{d}p_k(t)}{\mathrm{d}t} &= \lambda p_0(t) - (\lambda+\mu)p_k(t) + \mu p_{k+1}(t) \\
\frac{\mathrm{d}p_{k+1}(t)}{\mathrm{d}t} &= \lambda p_1(t) - (\lambda+\mu)p_{k+1}(t) + \mu p_{k+2}(t)
\end{aligned} \tag{A.2}$$

etc.

Finding the Equilibrium Solution

To find the equilibrium solution in which $p_n(t) \rightarrow p_n$ and $\frac{\mathrm{d}p_n(t)}{\mathrm{d}t} \rightarrow 0$ as $t \rightarrow 0$, we rearrange A.2 to give A.3.

$$\begin{aligned} -\lambda p_0 + \mu p_1 &= 0 \\ -(\lambda+\mu)p_1 + \mu p_2 &= 0 \\ &\vdots \\ -(\lambda+\mu)p_{k-1} + \mu p_k &= 0 \\ \lambda p_0 - (\lambda+\mu)p_k + \mu p_{k+1} &= 0 \\ \lambda p_1 - (\lambda+\mu)p_{k+1} + \mu p_{k+2} &= 0 \\ &\text{etc} \end{aligned} \tag{A.3}$$

By defining the generating function for p_n to be:

$$P(z) = \sum_{n=0}^{\infty} p_n z^n$$

and then multiplying the Equations in A.3 in turn by $z^0, z^1, \cdots, z^{k-1}, z^k, \cdots$ (respectively) and adding together gives us the z-transform shown in A.4.

$$\begin{aligned} & \lambda z^k \sum_{n=0}^{\infty} p_n z^n - \lambda \sum_{n=0}^{\infty} p_n z^n - \mu \sum_{n=1}^{\infty} p_n z^n + \mu \sum_{n=1}^{\infty} p_n z^{n-1} = 0 \\ \text{therefore} \quad & \lambda z^k P(z) - \lambda P(z) - \mu(P(z) - p_0) + \frac{\mu(P(z) - p_0)}{z} = 0 \\ \text{and} \quad & P(z)[\lambda z^{k+1} - (\lambda+\mu)z + \mu] = \mu p_0(1-z) \\ \text{finally} \quad & P(z) = \frac{\mu p_0(1-z)}{\lambda z^{k+1} - (\lambda+\mu)z + \mu} \end{aligned} \tag{A.4}$$

We can see that the equations for the the single-server queue with single arrivals (see Kleinrock (1975) for more information) is simply a special case of the Equations given in A.4. If we take $P(1) = 1$ and use L'Hopital's rule (to avoid the indeterminate form that would arise if we simply substituted $z = 1$) then we get the Equations in A.5:

$$
\begin{aligned}
P(1) &= \lim_{z \to 1} \frac{-\mu p_0}{(k+1)\lambda z^k - (\lambda + \mu)} \\
&= \frac{-\mu p_0}{k\lambda - \mu} \\
&= \frac{\mu p_0}{\mu - k\lambda} \\
&= 1
\end{aligned}
$$

therefore

$$
\begin{aligned}
p_0 &= \frac{\mu - k\lambda}{\mu} \\
&= 1 - \frac{k\lambda}{\mu}
\end{aligned}
\tag{A.5}
$$

And, since p_0 must be positive, we have $\frac{k\lambda}{\mu} < 1$ as a necessary condition for stability. This is simply equivalent to $\mu > k\lambda$, which states that the rate of service of requests must be greater than the arrival rate of requests.

$$
\begin{aligned}
P(z) &= \frac{(1 - \frac{k\lambda}{\mu})(1 - z)}{1 - (1 + \frac{\lambda}{\mu})z + \frac{\lambda}{\mu} z^{k+1}} \\
&= \frac{(1 - \frac{k\lambda}{\mu})(1 - z)}{1 - z - \frac{\lambda}{\mu} z(1 - z^k)} \\
&= \frac{(1 - \frac{k\lambda}{\mu})}{1 - \frac{\lambda}{\mu} z(\frac{1 - z^k}{1 - z})} \\
&= \frac{(1 - \frac{k\lambda}{\mu})}{1 - \frac{\lambda}{\mu} z(1 + z + z^2 + \cdots + z^{k-1})} \\
&= \frac{(1 - \frac{k\lambda}{\mu})}{1 - \frac{\lambda}{\mu} z - \frac{\lambda}{\mu} z - \frac{\lambda}{\mu} z^2 - \cdots - \frac{\lambda}{\mu} z^k}
\end{aligned}
\tag{A.6}
$$

Now, to find an expression for the average number of requests in the system, we have to take the first derivative of the above z-transform (Equation A.6) evaluated at $z = 1$. This is shown in Equation A.7.

$$P'(z) = \frac{\mathrm{d}P}{\mathrm{d}z} = \sum_{n=0}^{\infty} np_n z^{n-1}$$

and

$$P'(z)\big|_{z=1} = \sum_{n=0}^{\infty} np_n$$

so

$$P'(z) = \frac{\frac{\lambda}{\mu}(1-k\frac{\lambda}{\mu})(1+2z+3z^2+\cdots+kz^{k-1})}{[1-k\frac{\lambda}{\mu}z-k\frac{\lambda}{\mu}z^2-k\frac{\lambda}{\mu}z^k]^2}$$

therefore

$$P'(1) = \frac{\frac{\lambda}{\mu}(1-k\frac{\lambda}{\mu})(1+2+3+\cdots+k)}{[1-k\frac{\lambda}{\mu}z-k\frac{\lambda}{\mu}z^2-k\frac{\lambda}{\mu}z^k]^2}$$

and

$$= \frac{k(k+1)\frac{\lambda}{\mu}}{2(1-k\frac{\lambda}{\mu})} \tag{A.7}$$

The expression for the number of requests in the system is therefore:

$$L = \frac{k(k+1)\frac{\lambda}{\mu}}{2(1-k\frac{\lambda}{\mu})} \tag{A.8}$$

Appendix B

Rearranging Equation 4.6

The mean response time of the system, $\overline{T}_{sys}$, is given in Equation B.1.

$$\overline{T}_{sys} = \sum_{i=1}^{n} \alpha_i \cdot \frac{\alpha_i k + 1}{2\mu - 2\alpha_i k\lambda} \tag{B.1}$$

Multiplying the top and bottom of Equation B.1 by -2λ gives Equation B.2.

$$\overline{T}_{sys} = \frac{1}{-2\lambda} \cdot \sum_{i=1}^{n} \alpha_i \cdot \frac{-2\lambda\alpha_i k - 2\lambda}{2\mu - 2\alpha_i k\lambda} \tag{B.2}$$

Adding $+2\mu - 2\mu$ to the numerator of Equation B.2 gives Equation B.3

$$\overline{T}_{sys} = \frac{1}{-2\lambda} \cdot \sum_{i=1}^{n} \alpha_i \cdot \frac{-2\lambda\alpha_i k - 2\lambda + 2\mu - 2\mu}{2\mu - 2\alpha_i k\lambda} \tag{B.3}$$

Cancelling terms in Equation B.3 and simplifying results in Equation B.4.

$$\overline{T}_{sys} = \frac{1}{-2\lambda} \cdot \sum_{i=1}^{n} \alpha_i \cdot \frac{\cancel{-2\lambda\alpha_i k} - 2\lambda + \cancel{2\mu} - 2\mu}{2\mu - 2\alpha_i k\lambda}$$

$$\overline{T}_{sys} = \frac{1}{-2\lambda} \cdot \sum_{i=1}^{n} \alpha_i \cdot \left[1 - \frac{2\lambda + 2\mu}{2\mu - 2\alpha_i k\lambda} \right]$$

$$\overline{T}_{sys} = \frac{1}{-2\lambda} \cdot \sum_{i=1}^{n} \left[\alpha_i - \frac{2\lambda\alpha_i + 2\mu\alpha_i}{2\mu - 2\alpha_i k\lambda} \right] \tag{B.4}$$

Appendix C

Solving Equation 4.9 Using Lagrange Multipliers

The Lagrangian function for Equation 4.9 is given by:

$$h(\alpha, \phi) = F(\alpha) - \phi \cdot \left[\sum_{i=1}^{n} \alpha_i - 1 \right] \tag{C.1}$$

where ϕ is the Lagrange multiplier. Setting the partial derivatives of Equation C.1 to zero gives:

$$\frac{\partial h}{\partial \alpha_i} = \frac{dF}{d\alpha_i} - \phi \cdot \frac{d}{d\alpha_i} \left[\sum_{i=1}^{n} \alpha_i - 1 \right] = 0 \tag{C.2}$$

Taking the derivative of the first term in Equation C.2 gives:

$$\frac{dF}{d\alpha_i} = \frac{d}{d\alpha_i} \left[\alpha_i - \frac{2\lambda\alpha_i + 2\mu_i\alpha_i}{2\mu_i - 2\alpha_i k\lambda} \right] \tag{C.3}$$

Which reduces to:

$$\frac{dF}{d\alpha_i} = 1 - \frac{d}{d\alpha_i}\left(\frac{u}{v}\right)$$

$$\text{where} \quad u = 2\lambda\alpha_i + 2\mu_i\alpha_i \quad \text{and} \quad u' = 2\lambda + 2\mu_i \qquad \text{(C.4)}$$
$$v = 2\mu_i - 2\alpha_i k\lambda \qquad v' = -2\lambda k$$

Using the quotient rule gives:

$$\frac{d}{d\alpha_i}\left(\frac{u}{v}\right) = \frac{vu' - uv'}{v^2}$$

$$= \frac{(2\mu_i - 2\lambda\alpha_i k)(2\lambda + 2\mu_i) - (2\lambda\alpha_i + 2\mu_i\alpha_i)(-2\lambda k)}{(2\mu_i - 2\lambda\alpha_i k)^2}$$

$$= \frac{4\mu_i\lambda - \cancel{4\lambda^2\alpha_i k} + 4\mu_i^2 - \cancel{4\mu_i\lambda\alpha_i k} + \cancel{4\lambda^2\alpha_i k} + \cancel{4\mu_i\lambda\alpha_i k}}{(2\mu_i - 2\lambda\alpha_i k)^2}$$

$$= \frac{4\mu_i\lambda + 4\mu_i^2}{(2\mu_i - 2\lambda\alpha_i k)^2}$$

Therefore Equation C.3 becomes:

$$\frac{dF}{d\alpha_i} = 1 - \frac{4\mu_i\lambda + 4\mu_i^2}{(2\mu_i - 2\lambda\alpha_i k)^2} \qquad \text{(C.5)}$$

Taking the derivative of the second term in Equation C.2 gives:

$$\frac{d}{d\alpha_i}\left[\sum_{i=1}^{n}\alpha_i - 1\right] = n \qquad \text{(C.6)}$$

Therefore Equation C.2 reduces to:

$$\frac{\partial h}{\partial \alpha_i} = 1 - \frac{4\mu_i\lambda + 4\mu_i^2}{(2\mu_i - 2\lambda\alpha_i k)^2} - n\phi = 0 \qquad \text{(C.7)}$$

Solving Equation C.7 for α_i results in:

$$\begin{aligned}
\frac{4\mu_i\lambda + 4\mu_i^2}{(2\mu_i - 2\lambda\alpha_i k)^2} &= (1 - n\phi) \\
4\mu_i\lambda + 4\mu_i^2 &= (1 - n\phi)(2\mu_i - 2\lambda\alpha_i k)^2 \\
\sqrt{4\mu_i\lambda + 4\mu_i^2} &= \sqrt{(1 - n\phi)} \cdot (2\mu_i - 2\lambda\alpha_i k) \qquad \text{(C.8)} \\
\frac{\sqrt{4\mu_i\lambda + 4\mu_i^2}}{\sqrt{(1 - n\phi)}} &= 2\mu_i - 2\lambda\alpha_i k \\
\alpha_i &= \left[2\mu_i - \frac{\sqrt{4\mu_i\lambda + 4\mu_i^2}}{\sqrt{(1 - n\phi)}}\right] \cdot \frac{1}{2\lambda k}
\end{aligned}$$

Substituting Equation C.8 into the constraint gives:

$$\begin{aligned}
\textstyle\sum_{i=1}^{n} \frac{1}{2\lambda k}\left[2\mu_i - \frac{\sqrt{4\mu_i\lambda + 4\mu_i^2}}{\sqrt{(1 - n\phi)}}\right] &= 1 \\
\textstyle\sum_{i=1}^{n} \left[2\mu_i - \frac{\sqrt{4\mu_i\lambda + 4\mu_i^2}}{\sqrt{(1 - n\phi)}}\right] &= 2\lambda k \\
\textstyle\sum_{i=1}^{n} 2\mu_i - \sum_{i=1}^{n} \frac{\sqrt{4\mu_i\lambda + 4\mu_i^2}}{\sqrt{(1 - n\phi)}} &= 2\lambda k \qquad \text{(C.9)} \\
\textstyle\sum_{i=1}^{n} \frac{\sqrt{4\mu_i\lambda + 4\mu_i^2}}{\sqrt{(1 - n\phi)}} &= \textstyle\sum_{i=1}^{n} 2\mu_i - 2\lambda k \\
\frac{1}{\sqrt{(1 - n\phi)}} \cdot \sum_{i=1}^{n} \sqrt{4\mu_i\lambda + 4\mu_i^2} &= \textstyle\sum_{i=1}^{n} 2\mu_i - 2\lambda k \\
\frac{1}{\sqrt{(1 - n\phi)}} &= \frac{\sum_{i=1}^{n} 2\mu_i - 2\lambda k}{\sum_{i=1}^{n} \sqrt{4\mu_i\lambda + 4\mu_i^2}}
\end{aligned}$$

Finally, substituting Equation C.9 into Equation C.8 gives:

$$\alpha_i = \frac{1}{2\lambda k} \cdot \left[2\mu_i - \sqrt{4\mu_i\lambda + 4\mu_i^2} \cdot \frac{\sum_{i=1}^{n} 2\mu_i - 2\lambda k}{\sum_{i=1}^{n} \sqrt{4\mu_i\lambda + 4\mu_i^2}}\right] \qquad \text{(C.10)}$$

References

Abboud, K. and Schoenauer, M.: 2002, Surrogate deterministic mutation: Preliminary results, *in* P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton and M. Schoenauer (eds), *Proceedings of the Fifth European Conference on Artificial Evolution (EA 2001)*, Vol. 2310 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 104–116.

Abdalhaq, B., Cortes, A., Margalef, T. and Luque, E.: 2002, Evolutionary optimization techniques on computational grids, *Proceedings of the International Conference on Computer Science (ICCS2002)*, Springer-Verlag, pp. 513–522.

Abido, M. A.: 2006, Multiobjective evolutionary algorithms for electric power dispatch problem, *IEEE Transactions on Evolutionary Computation* 10(3), 315–329.

Adra, S. F., Griffin, I. and Fleming, P. J.: 2007, A comparative study of progressive preference articulation techniques for multiobjective optimisation, *in* S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu and T. Murata (eds), *Proceedings of the Fourth International Conference on Evolutionary Multi-criteria Optimisation (EMO) 2007*, Vol. 4403 of *Lecture Notes on Computer Science*, pp. 908–921.

Ahsant, M., Basney, J., Mulmo, O., Lee, A. J. and Johnsson, L.: 2006, Toward an on-demand restricted delegation mechanism for grids, *The 7th IEEE/ACM International Conference on Grid Computing*, pp. 152–159.

Alander, J. T.: 2003, Indexed bibliography of distributed genetic algorithms, *Technical Report 94-1-PARA*, University of Vaasa.

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I. and Angel, S.: 1977, *A Pattern Language: Towns, Buildings, Construction*, Oxford University Press, Oxford.

Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D. and Tuecke, S.: 2002, Data management and transfer in high-performance computational grid environments, *Parallel Computing* **28**(5), 749–771.

Anderson, S. R., Kadirkamanathan, V., Chipperfield, A., Sharifi, V. and Swithenbank, J.: 2005, Multi-objective optimization of operational variables in a waste incineration plant, *Computers and Chemical Engineering* **29**, 1121–1130.

Aouni, B. and Kettani, O.: 2001, Goal programming model: A glorious past and a promising future, *European Journal of Operations Research* **133**, 225–231.

Argyle, J. P. M.: 2006, *Optimisation of Operational Cost with Application to an Aerospace Engine System*, PhD thesis, University of Sheffield.

Argyle, J. P. M. and Tubby, J.: 2002, Integrated logistics support optimisation, *Technical Report RRUTC/Shef/R/02006*, Rolls-Royce PLC.

Atkins, D., Graff, M., Lenstra, A. K. and Leyland, P. C.: 1994, The magic words are squeamish ossifrage, *Proceedings of ASIACrypt'94 - Advances in Cryptology*, Vol. 917 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 265–277.

Atkinson, M., Chervenak, A. L., Kunszt, P., Narang, I., Paton, N. W., Pearson, D., Shoshani, A. and Watson, P.: 2004, Data access, integration, and

management, *in* I. Foster and C. Kessleman (eds), *The GRID2: Blueprint for a New Computing Infrastructure*, second edition edn, Morgan Kaufmann, chapter 22, pp. 391–430.

ATKOSoft: 1997, Survey of visualisation methods and software tools. `http://europa.eu.int/en/comm/eurostat/research/\\supcom.96/30/result/a/visualisation\_methods.pdf`.

Austin, J., Jackson, T., Fletcher, M., Jessop, M., Cowley, P. and Lobner, P.: 2004, Predictive maintenance: Distributed aircraft engine diagnostics, *in* I. Foster and C. Kessleman (eds), *The GRID2: Blueprint for a New Computing Infrastructure*, second edition edn, Morgan Kaufmann, chapter 5, pp. 69–79.

Axelrod, R.: 1984, *The Evolution of Cooperation*, Basic Books, New York.

Axelrod, R.: 1987, The evolution of strategies in the iterated prisoner's dilemma, *in* L. Davis (ed.), *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann, pp. 32–41.

Babu, B. V., Angira, R. and Nilekar, A.: 2004, Optimal design of an auto-thermal ammonia synthesis reactor using differential evolution, *Proceedings of Systemics, Cybernetics and Informatics (SCI2004)*, IIIS Press.

Bäck, T.: 1992, The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm, *in* R. Männer and B. Manderick (eds), *Parallel Problem Solving from Nature 2*, North-Holland, Amsterdam, pp. 85–94.

Bäck, T.: 1996, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, Oxford.

Bäck, T., Hammel, U. and Schwefel, H.-P.: 1997, Evolutionary computation: Comments on the history and current state, *IEEE Transactions on Evolutionary Computation* **1**(1), 3–17.

Bäck, T., Hoffmeister, F. and Schwefel, H.-P.: 1991, A survey of evolution strategies, *in* R. K. Belew and L. B. Booker (eds), *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 2–9.

Baker, J. E.: 1985, Adaptive selection methods for genetic algorithms, *in* J. J. Grefenstette (ed.), *Proceedings of the First International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, New Jersey, pp. 101–111.

Baker, J. E.: 1987, Reducing bias and inefficiency in the selection algorithm, *in* J. J. Grefenstette (ed.), *Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, New Jersey, pp. 14–21.

Baker, M., Buyya, R. and Laforenza, D.: 2002, Grid and grid technologies for wide area distributed computing, *Software: Practice and Experience* **32**(15), 1437–1466.

Banawan, S. A. and Zeidat, N. M.: 1992, A comparative study of load sharing in heterogeneous multicomputer systems, *Proceedings of the 25th Annual Symposium on Simulation*, IEEE Computer Society Press, pp. 22–31.

Barnard, S., Biswas, R., Saini, S., Van der Wijngaart, R., Yarrow, M., Zechter, L., Foster, I. and Larsson, O.: 1999, Large scale computational fluid dynamics on the Information Power Grid using globus, *Proceedings of the The 7th Symposium on the Frontiers of Massively Parallel Computation*, pp. 60–67.

Barricelli, N. A.: 1962, Numerical testing of evolution theories: Part I, *ACTA Biotheoretica* **16**, 69–98.

Baru, C., Moore, R., Rajasekar, A. and Wan, M.: 1998, The sdsc storage resource broker, *in* S. A. MacKay and J. H. Johnson (eds), *Proceedings of the International Conference of the Centre for Advanced Studies on Collaborative Research (CASCON) 1998*, IBM Press.

Benedetti, A., Farina, M. and Gobbi, M.: 2006, Evolutionary multiobjective industrial design: The case of a racing car tire-suspension system, *IEEE Transactions on Evolutionary Computation* **10**(3), 230–244.

Berman, F.: 1999, High-performance schedulers, *in* I. Foster and C. Kesselman (eds), *The GRID: Blueprint for a New Computing Infrastructure*, first edition edn, Morgan Kaufmann, chapter 12, pp. 279–309.

Berman, F., Wolski, R., Figueira, S., Schopf, J. and Shao, G.: 1996, Application-level scheduling on distributed heterogeneous networks, *Supercomputing '96*.

Berners-Lee, T.: 1999, *Weaving the Web*, Harper San Francisco.

Bertsekas, D. P.: 1982, *Constrained Optimisation and Lagrange Multiplier Methods*, Academic Press.

Beyer, H.-G., O'Reilly, U.-M., Arnold, D. V., Banzhaf, W., Blum, C., Bonabeau, E. W., Paz, E. C., Dasgupta, D., Deb, K., Foster, J. A., de Jong, E. D., Lipson, H., Llora, X., Mancoridis, S., Pelikan, M., Raidl, G. R., Soule, T., Tyrrell, A., Watson, J.-P. and Zitzler, E. (eds): 2005, *Proceedings of the Genetic and Evolutionary Computation COnference (GECCO) 2005*, ACM Press, New York.

Beyer, H.-G. and Schwefel, H.-P.: 2002, Evolution strategies: A comprehensive introduction, *Natural Computing* **1**, 35–52.

Biles, J. A.: 2003, Genjam in perspective: A tentative taxonomy for GA music and art systems, *Leonardo* 36(1), 43–45.

Blickle, T. and Thiele, L.: 1995, A comparison of selection schemes used in genetic algorithms, *TIK-Report 11*, TIK Institut fur Technische Informatik und Kommunikationsnetze, Computer Engineering and Networks Laboratory, ETH, Swiss Federal Institute of Technology.

Bosman, P. A. N. and Thierens, D.: 2003, The balance between proximity and diversity in multiobjective evolutionary algorithms, *IEEE Transactions on Evolutionary Computation* 7(2), 174–188.

Box, G. E. P.: 1957, Evolutionary operation: A method for increasing industrial productivity, *Applied Statistics* 6(2), 81–101.

Branke, J. and Deb, K.: 2004, Integrating user preferences into evolutionary multi-objective optimization, *in* Y. Jin (ed.), *Knowledge Incorporation in Evolutionary Computation*, Springer, Berlin, pp. 461–477.

Branke, J., Kaußler, T. and Schmeck, H.: 2001, Guidance in evolutionary multi-objective optimization, *Advances in Engineering Software* 32, 499–507.

Brans, J. P., Vincke, P. and Mareschal, B.: 1986, How to select and how to rank projects: The PROMETHEE method, *European Journal of Operations Research* 24, 228–238.

Brooke, J. M., Coveney, P. V., Harting, J., Jha, S., Pickles, S. M., Pinning, R. L. and Porter, A. R.: 2003, Computational steering in realitygrid, *in* S. Cox (ed.), *Proceedings of the U.K. e-Science All Hands Meeting*.

Brunett, S., Czajkowski, K., Fitzgerald, S., Kesselman, C., Foster, I., Tuecke, S., Johnson, A. and Leigh, J.: 1998, Application experiences with the Globus Toolkit, *Proceedings of the High Performance Distributed Computing Symposium (HPDC '98)*, IEEE Computer Society, pp. 81–88.

Brunett, S., Davis, D., Gottschalk, T., Messina, P. and Kesselman, C.: 1998, Implementing distributed synthetic forces simulations in metacomputing environments, *Proceedings of the Heterogeneous Computing Workshop*, pp. 29–42.

Buchtala, O., Klimek, M. and Sick, B.: 2005, Evolutionary optimization of radial basis function classifiers for data mining applications, *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics* **35**(5), 928–947.

Bull, L.: 1999, On model-based evolutionary computation, *Soft Computing* **3**, 76–82.

Bullock, S., Cartlidge, J. and Thompson, M.: 2002, Prospects for computational steering in evolutionary computation, *in* E. Bilotta, D. Groß, T. Smith, T. Lenaerts, S. Bullock, H. H. Lund, J. Bird, R. Watson, P. Pantano, L. Pagliarini, H. Abbass, R. Standish and M. Bedau (eds), *Artificial Life VIII Workshop Proceedings*, MIT Press, pp. 131–137.

Burress, J. R., Abla, G., Flanagan, S., Keahey, K., Leggett, T., Ludescher, C., McCune, D., Papka, M. E., Peng, Q., Randerson, L. and Schissel, D. P.: 2005, Developments in remote collaboration and computation, *Fusion Science and Technology* **47**(3), 814–818.

Cantú-Paz, E.: 1999, *Designing Efficient and Accurate Parallel Genetic Algorithms*, PhD thesis, Illinois Genetic Algorithms Laboratory.

Cantú-Paz, E. and Goldberg, D. E.: 1999, On the scalability of parallel genetic algorithms, *Evolutionary Computation* **7**(4), 429–449.

Carr, D. B., Nicholson, W. L., Littlefield, R. J. and Hall, D. L.: 1986, Interactive color display methods for multivariate data, *in* E. J. Wegman and D. J. DePriest (eds), *Statistical Image Processing*, Dekker, New York, pp. 215–250.

Casavant, T. L. and Kuhl, J. G.: 1988, A taxonomy of scheduling in general-purpose distributed computing systems, *IEEE Transactions on Software Engineering* **14**(2), 141–154.

Casey, L. M.: 1981, Decentralized scheduling, *Australian Computer Journal* **13**, 58–63.

Cerf, V., Dalal, Y. and Sunshine, C.: 1974, RFC675: Specification of the Internet Transmission Control Program. Available from `http://www.ietf.org/rfc/rfc675.txt`.

Chambers, J. M., Cleveland, W. S. and Tukey, P. A.: 1983, *Graphical methods for data analysis*, Duxbury Press.

Chanson, S. T., Deng, W., Hui, C.-C., Tang, X. and To, M. Y.: 2000, Multidomain load balancing, *Proceedings of the 2000 International Conference on Network Protocols (ICNP '00)*, pp. 315–324.

Chappell, D. A. and Jewell, T.: 2002, *Java Web Services*, O'Reilly.

Chellapilla, K. and Fogel, D. B.: 2001, Evolving an expert checkers playing program without using human expertise, *IEEE Transactions on Evolutionary Computation* **5**(4), 422–428.

Chen, W., Allen, J., Mavris, D. and Mistree, F.: 1996, A concept exploration method for determining robust top-level specifications, *Engineering Optimization* **26**, 137–158.

Chetty, M. and Buyya, R.: 2002, Weaving computational grids: How analogous are they with electrical grids?, *Computing in Science and Engineering* **4**(4), 61–71.

Chin, J., Harting, J., Jha, S., Coveney, P. V., Porter, A. R. and Pickles, S. M.: 2003, Steering in computational science: Mesoscale modelling and simulation, *Contemporary Physics* 44(5), 417–434.

Chipperfield, A. J. and Fleming, P. J.: 1995, Parallel genetic algorithms, *in* A. Y. Zomaya (ed.), *Parallel And Distributed Computing Handbook*, McGraw-Hill, chapter 39, pp. 1118–1144.

Chipperfield, A. J., Fleming, P. J., Pohlheim, H. and Fonseca, C. M.: 1994, Genetic algorithm toolbox user's guide, *Technical Report 512*, University of Sheffield.

ClimatePrediction.net project: 2006, ClimatePrediction.net website. Viewed 28 August 2006.
URL: *http://climateprediction.net/*

Cluster Resources Inc.: 2004, MOAB grid scheduler (silver). Viewed 22nd April 2007.
URL: *http://www.supercluster.org/projects/silver/*

Coello, C. A. C., Aguirre, A. H. and Zitzler, E. (eds): 2005, *Proceedings of the Third International Conference on Evolutionary Multi-Criteria Optimisation (EMO2005)*, Vol. 3410 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin.

Coello, C. A. and Pulido, G. T.: 2001, Multiobjective optimization using a micro genetic algorithm, *in* L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon and E. Burke (eds), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2001*, Morgan Kaufmann, pp. 274–282.

Coello-Coello, C. A.: 2000a, Handling preferences in evolutionary multiobjective optimization: A survey, *Proceedings of the Congress on Evolutionary Computation (CEC) 2000*, IEEE Press, pp. 30–37.

Coello-Coello, C. A.: 2000b, Treating constraints as objectives for single-objective evolutionary optimization, *Engineering Optimization* **32**(3), 275–308.

Coello-Coello, C. A.: 2002, Theoretical and numerical constraint-handling techniques used with evoluionary algorithms : a survey of the state of the art., *Computer Methods in Applied Mechanics and Engineering* **191**(11), 1245–1287.

Coello-Coello, C. A.: 2006, Evolutionary multi-objective optimization: A historical view of the field, *IEEE Computational Intelligence Magazine* pp. 28–36.

Coello-Coello, C. A., Van Veldhuizen, D. A. and Lamont, G. B.: 2002, *Evolutionary Algorithms for Solving Multiobjective Problems*, Kluwer, London.

Cohon, J. L. and Marks, D. H.: 1975, Evaluation of multiobjective programming techniques, *Water Resources Research* **11**(2), 208–229.

Colajanni, M., Yu, P. S. and Cardellini, V.: 1998, Dynamic load balancing in geographically distributed heterogeneous web servers, *Proceedings of the 18th International Conference on Distributed Computing Systems*, pp. 295–302.

Colan, M.: 2004, Service oriented architecture expands the vision of web services: Part 1, *Developerworks paper*, IBM. Viewed 18 October 2006.
URL: *http://www-128.ibm.com/developerworks/library/ws-soaintro.html*

Cook, D., Hofmann, H., Lee, E.-K., Yang, H., Nikolau, B. and Wurtele, E.: 2007, Exploring gene expression data, using plots, *Journal of Data Science* **5**, 151–182.

Corbató, F. J. and Vyssotsky, V. A.: 1965, Introduction and overview of the multics system, *Proceedings of the Fall Joint Computer Conference*, Vol. 27, pp. 185–196. Available from `http://www.multicians.org/fjcc1.html`.

Corne, D. W., Knowles, J. D. and Oates, M. J.: 2000, The Pareto envelope-based selection algorithm for multiobjective optimization, *in* M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo and H.-P. Schwefel (eds), *Parallel Problem Solving from Nature VI*, Vol. 1917 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 839–848.

Courant, R.: 1943, Variational methods for the solution of problems of equilibrium and vibration, *Bulletin of the American Mathematical Society* **49**, 1–23.

Crocker, J. and Kumar, U. D.: 2000, Age-related maintenance versus reliability centred maintenance: A case study on aero-engines, *Reliability Engineering and Systems Safety* **67**, 113–118.

Cvetkovic, D. and Parmee, I. C.: 1999, Genetic algorithm-based multi-objective optimisation and conceptual engineering design, *Proceedings of the Congress on Evolutionary Computation (CEC) 1999*, IEEE Press, pp. 29–36.

Cvetkovic, D. and Parmee, I. C.: 2002, Preferences and their application in evolutionary multiobjective optimization, *IEEE Transactions on Evolutionary Computation* **6**(1), 42–57.

Czajkowski, K., Foster, I., Kesselman, C., Martin, S., Smith, W. and Tuecke, S.: 1998, A resource management architecture for metacomputing systems, *in* D. G. Feitelson and L. Rudolph (eds), *Job Scheduling Strategies for Parallel Processing - IPPS/SPDP'98 Workshop*, Vol. 1459 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 62–82.

Dantzig, G. B.: 1963, *Linear Programming and Extensions*, Princeton University Press, New Jersey.

Darwin, C.: 1859, *The Origin of Species*, John Murray, London.

Davis, L.: 1985, Job shop scheduling with genetic algorithms, *in* J. J. Grefenstette (ed.), *Proceedings of the First International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, New Jersey, pp. 136–140.

Davis, L. (ed.): 1991, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.

Dawkins, R.: 1976, *The Selfish Gene*, Oxford University Press, Oxford.

De Jong, K. A.: 1975, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, PhD thesis, University of Michigan.

Deb, K.: 1999a, Multi-objective evolutionary algorithms: Introducing bias among Pareto-optimal solutions, *KanGAL Report 99002*, Kanpur Genetic Algorithms Laboratory (KanGAL).

Deb, K.: 1999b, Multi-objective genetic algorithms: Problem difficulties and construction of test problems, *Evolutionary Computation* **7**(3), 205–230.

Deb, K.: 2001, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, New York.

Deb, K. and Goldberg, D. E.: 1989, An investigation of niche and species formation in genetic function optimization, *in* J. D. Schaffer (ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 42–50.

Deb, K. and Goyal, M.: 1996, A combined genetic adaptive search (geneas) for engineering design, *Computer Science and Informatics* **26**(4), 30–45.

Deb, K., Pratap, A., Agarwel, S. and Meyarivan, T.: 2002, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation* **6**(2), 182–197.

Deb, K., Pratap, A., S.Agarwel and Meyarivan, T.: 2000, A fast and elitist multi-objective genetic algorithm: NSGA-II, *in* M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo and H.-P. Schwefel (eds), *Parallel Problem Solving from Nature VI*, Vol. 1917 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 849–858.

Deb, K., Sundar, J., Bhaskara Rao, N. and Chaudhuri, S.: 2006, Reference point based multi-objective optimization using evolutionary algorithms, *International Journal of Computational Intelligence Research* **2**(3), 273–286.

Deb, K., Thiele, L., Laumanns, M. and Zitzler, E.: 2002, Scalable multi-objective optimization test problems, *Proceedings of the Congress on Evolutionary Computation (CEC) 2002*, IEEE Press, pp. 825–830.

Deb, K., Zope, P. and Jain, A.: 2003, Distributed computing of pareto optimal solutions with evolutionary algorithms, *Proceedings of the Second International Conference on Evolutionary Multi-Criteria Optimisation (EMO2003)*, Springer-Verlag, pp. 534–549.

DeFanti, T. A., Foster, I., Papka, M. E., Stevens, R. and Kuhfuss, T.: 1996, Overview of the i-way: Wide area visual supercomputing, *The International Journal of Supercomputer Applications and High Performance Computing* **10**(2/3), 123–131.

Dhar, V. and Ranganathan, N.: 1990, Integer programming vs. expert systems: An experimental comparison, *Artificial Intelligence and Language Processing* **33**(3), 323–336.

Distributed Systems Architecture Group, Universidad Complutense de Madrid: 2007, GridWay Metascheduler. Viewed 22nd April 2007.
URL: *http://www.gridway.org/index.php*

Distributed.net project: 2006, Distributed.net website. Viewed 28 August 2006.
URL: *http://distributed.net/*

Dongarra, J.: 2006, Trends in high-performance computing, *IEEE Circuits and Devices Magazine* pp. 22–27.

Duda, J. and Osyczka, A.: 2005, Multiple criteria lot-sizing in a foundry using evolutionary algorithms, *in* C. A. C. Coello, A. H. Aguirre and E. Zitzler (eds), *Proceedings of the Third International Conference on Evolutionary Multi-criteria Optimization (EMO) 2005*, Vol. 3410 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 651–663.

Edgeworth, F. Y.: 1932, *Mathematical Psychics. An Essay on the Application of Mathematics to the Moral Sciences*, Vol. 10 of *Reprints of scarce tracts in economic and political science*, reprint edn, London School of Economics and Political Science, London. Originally published : London : Kegan Paul ; 1881.

Eiben, A. E. and Schippers, C. A.: 1998, On evolutionary exploration and exploitation, *Fundamenta Informaticae* **35**, 1–16.

Epema, D. H. J., Livny, M., van Dantzig, R., Evers, X. and Pruyne, J.: 1996, A worldwide flock of condors: Load sharing among workstation clusters, *Future Generation Computer Systems* **12**, 53–65.

Erkmen, I., Erkmen, A. M. and Günver, H.: 2000, Robot hand preshaping and regrasping using genetic algorithms, *The International Journal of Robotics Research* **19**(9), 857–874.

Farina, M. and Amato, P.: 2002, On the optimal solution definition for many-criteria optimization problems, *in* J. Keller and O. Nasraoui (eds), *Proceedings of the NAFIPS-FLINT International Conference 2002*, IEEE Press, pp. 233–238.

Farina, M. and Amato, P.: 2004, A fuzzy definition of "optimality" for many-criteria optimization problems, *IEEE Transactions on System, Man and Cybernetics - Part A: Systems and Humans* **34**(3), 315–326.

Farrell, S. and Housley, R.: 2002, RFC3281:an Internet attribute certificate profile for authorization. Available from `http://www.ietf.org/rfc/rfc3281.txt`.

Fernandez, F., Tomassini, M. and Vanneschi, L.: 2003, An empirical study of multipopulation genetic programming, *Genetic Programming and Evolvable Machines* 4, 21–51.

Fisher, R. A.: 1930, *The Genetical Theory of Natural Selection*, Clarendon Press, Oxford.

Fleming, P. J., Purshouse, R. C. and Lygoe, R. J.: 2005, Many objective optimization: An engineering perspective, *in* C. A. C. Coello, A. H. Aguirre and E. Zitzler (eds), *Proceedings of the International Conference on Evolutionary Multi-Objective Optimization (EMO2005)*, Vol. 3470 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 14–32.

Fletcher, R.: 1981, *Practical Methods of Optimization, Volume 2: Constrained Optimization*, John Wiley & Sons, New York.

Fogarty, T. C. and Huang, R.: 1991, Implementing the genetic algorithm on transputer based parallel processing systems, *in* H.-P. Schwefel and R. Männer (eds), *Parallel Problem Solving from Nature 1*, Vol. 496 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 145–149.

Fogel, D. B. and Ghoziel, A.: 1997, A note on representations and variation operators, *IEEE Transactions on Evolutionary Computation* **1**(2), 159–161.

Fogel, L. J., Owens, A. J. and Walsh, M. J.: 1966, *Artificial Intelligence Through Simulated Evolution*, John Wiley & Sons, New York.

Fonseca, C. M. and Fleming, P. J.: 1993, Genetic algorithms for multi-objective optimization: Formulation, discussion and generalization, *in* S. Forrest (ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 416–423.

Fonseca, C. M. and Fleming, P. J.: 1995, Multiobjective genetic algorithms made easy: Selection, sharing, and mating restriction, *in* A. M. S. Zalzala (ed.), *Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*, pp. 42–52.

Fonseca, C. M. and Fleming, P. J.: 1998, Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Part I: A unifed formulation, *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* **28**(1), 26–37.

Foster, I.: 2000, Internet computing and the emerging grid, *Nature Web Matters*. Available from `http://www.nature.com/nature/webmatters/grid/grid.html`.

Foster, I.: 2002a, The grid: A new infrastructure for 21st century science, *Physics Today* **55**(2), 42–47.

Foster, I.: 2002b, What is the grid?, *GRIDtoday* **1**(6). Viewed 22 March 2007.
URL: *http://www.gridtoday.com/02/0722/100136.html*

Foster, I.: 2005, Globus Toolkit version 4: Software for service-oriented systems, *in* H. Jin, D. Reed and W. Jiang (eds), *Proceedings of the IFIP International*

Conference on Network and Parallel Computing (NPC'05), Vol. 3779 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 2–13.

Foster, I., Geisler, J., Nickless, W., Smith, W. and Tuecke, S.: 1996, Software infrastructure for the i-way high-performance distributed computing experiment, *Proceedings of the High Performance Distributed Computing Symposium (HPDC '96)*, pp. 562–571.

Foster, I. and Kesselman, C.: 1997, Globus: A metacomputing infrastructure toolkit, *International Journal of Supercomputer Applications* **11**(2), 115–128.

Foster, I. and Kesselman, C.: 1999, The Globus Toolkit, *in* I. Foster and C. Kesselman (eds), *The GRID: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, chapter 11, pp. 259–278.

Foster, I., Kesselman, C., Nick, J. M. and Tuecke, S.: 2002, Grid services for distributed system integration, *IEEE Computer* **35**(6), 37 – 46.

Foster, I., Kesselman, C., Tsudik, G. and Tuecke, S.: 1998, A security architecture for computational grids, *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pp. 83–92.

Foster, I., Kesselman, C. and Tuecke, S.: 2001, The anatomy of the grid: Enabling scalable virtual organizations, *International Journal of Supercomputer Applications* **15**(3), 200–222.

Fourman, M. P.: 1985, Compaction of symbolic layout using genetic algorithms, *in* J. J. Grefenstette (ed.), *Proceedings of the First International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, New Jersey, pp. 141–153.

Frey, J., Tannenbaum, T., Livny, M., Foster, I. and Tuecke, S.: 2001, Condor-G: A computation management agent for multi-institutional grids, *Proceedings*

of the 10th IEEE Symposium on High Performance Distributed Computing (HPDC10).

Friedberg, R. M.: 1958, A learning machine: Part I, *IBM Journal* **2**(1), 2–13.

Friedberg, R. M., Dunham, B. and North, J. H.: 1959, A learning machine: Part 2, *IBM Journal* **3**(7), 282–287.

Fung, C. C., Li, J. B., Wong, K. W. and Wang, K. P.: 2004, A java-based parallel platform for the implementation of evolutionary computation for engineering applications, *International Journal of Systems Science* **35**(13-14), 741–750.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: 1995, *Design Patterns - Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, MA.

Giacoman Zarzar, M.: 2007, h_∞ *controller design for the lateral dynamics of a boeing 747-200*, Master's thesis, Instituto Technologico y de Estudios Superiores de Monterrey.

Gibson, E. and Burger, J.: 2003, Parallel genetic algorithms: An exploration of weather prediction through clustered computing, *Journal of Computing Sciences in Colleges* **18**(5), 272–273.

Glover, K. and Doyle, J. C.: 1988, State-space formulae for all stabilizing controllers that satisfy an h_∞ norm bound and relations to risk sensitivity, *Systems and Control letters* pp. 167–172.

Goldberg, D. E.: 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.

Goldberg, D. E.: 1991, Real-coded genetic algorithms, virtual alphabets and blocking, *Complex Systems* **5**, 139–167.

Goldberg, D. E. and Deb, K.: 1991, A comparative analysis of selection schemes used in genetic algorithms, *in* G. J. E. Rawlins (ed.), *Foundations of Genetic Algorithms*, Morgan Kaufmann, pp. 69–93.

Goldberg, D. E. and Richardson, J.: 1987, Genetic algorithms with sharing for multimodal function optimization, *in* J. J. Grefenstette (ed.), *Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, New Jersey, pp. 41–49.

Gorges-Schleuter, M.: 1989, ASPARAGOS an asynchronous parallel genetic optimization strategy, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA3)*, pp. 422–427.

Greenwood, G. W., Hu, X. and D'Ambrosio, J. G.: 1996, Fitness functions for multiple objective optimization problems: Combining preferences with Pareto rankings, *in* R. K. Belew and M. D. Vose (eds), *Foundations of Genetic Algorithms 4*, Morgan Kaufmann, pp. 437–456.

Grosso, P. B.: 1985, *Computer Simulation of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model*, PhD thesis, University of Michigan.

Hadj-Alouane, A. B. and Bean, J. C.: 1997, A genetic algorithm for the multiple-choice integer program, *Operations Research* 45(1), 92–101.

Hancock, P. J. B.: 1994, An empirical comparison of selection methods in evolutionary algorithms, *in* T. C. Fogarty (ed.), *Evolutionary Computing - AISB Workshop*, Vol. 865 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 80–94.

Hartl, D. L. and Clark, A. G.: 1997, *Principles of Population Genetics*, 3rd edn, Sinauer.

He, L., Jarvis, S. A., Spooner, D. P., Jiang, H., Dillenberger, D. N. and Nudd, G. R.: 2006, Allocating non-real-time and soft real-time jobs in multiclusters, *IEEE Transactions on Parallel and Distributed Systems* **17**(2), 99–112.

Henderson, R. L.: 1995, Job scheduling under the Portable Batch System, *in* D. G. Feitelson and L. Rudolph (eds), *Job Scheduling Strategies for Parallel Processing (IPPS Workshop)*, Vol. 949 of *LNCS*, Springer, Berlin, pp. 279–294.

Herrara, F., Lozano, M. and Verdegay, J. L.: 1998, Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis, *Artificial Intelligence Review* **12**, 265–319.

Hiroyasu, T., Miki, M. and Watanabe, S.: 2000, The new model of parallel genetic algorithm in multi-objective optimization problems - divided range multi-objective genetic algorithm, *Proceedings of the Congress on Evolutionary Computation (CEC) 2000*.

Holland, J. H.: 1975, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

Horn, J.: 1997, Multicriteria decision making, *in* T. Bäck, D. B. Fogel and Z. Michalewicz (eds), *Handbook of Evolutionary Computation*, IOP Publishing, Bristol, pp. F1.9:1 – F1.9:15.

Horn, J., Nafpliotis, N. and Goldberg, D. E.: 1994, A niched Pareto genetic algorithm for multiobjective optimization, *Proceedings of the Congress on Evolutionary Computation (CEC) 1994*, IEEE Press, pp. 82–87.

Hornik, K., Stinchcombe, M. and White, H.: 1989, Multilayer feedforward networks are universal approximators, *Neural Networks* **2**(5), 359–366.

IEE: 2005, *Proceedings of the Congress on Evolutionary Computation (CEC) 2005*, IEEE Press.

Inselberg, A.: 1985, The plane with parallel coordinates, *The Visual Computer* **1**, 69–91.

Jakob, W., Gorges-Schleuter, M. and Blume, C.: 1992, Application of genetic algorithms to task planning snd learning, *in* R. Männer and B. Manderick (eds), *Parallel Problem Solving from Nature 2*, North-Holland, Amsterdam, pp. 291–300.

Jensen, M. T.: 2003, Generating robust and flexible job shop schedules using genetic algorithms, *IEEE Transactions on Evolutionary Computation* **7**(3), 275–288.

Jin, R., Chen, W. and Simpson, T. W.: 2001, Comparative studies of metamodelling techniques under multiple modelling criteria, *Structural and Multidisciplinary Optimization* **23**(1), 1–13.

Jin, Y.: 2005, A comprehensive survey of fitness approximation in evolutionary computation, *Soft Computing* **9**, 3–12.

Jin, Y., Olhofer, M. and Sendhoff, B.: 2000, On evolutionary optimization with approximate fitness function, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2000*, Morgan Kaufmann, pp. 786–792.

Johanson, B. and Poli, R.: 1998, GP-Music: an interactive genetic programming system for music generation with automated fitness raters, *in* J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba and R. Riolo (eds), *Proceedings of the Third Annual Conference on Genetic Programming*, pp. 181–186.

Johnson, C. R., MacLeod, R., Parker, S. G. and Weinstein, D.: 2004, Biomedical computing and visualization software environments, *Communications of the ACM* **47**(11), 64–71.

Johnson, C. R., Parker, S. G., Hansen, C., Kindlmann, G. L. and Livnat, Y.: 1999, Interactive simulation and visualization, *IEEE Computer* **32**(12), 59–65.

Karasavvas, K., Antonioletti, M., Atkinson, M. P., Chue Hong, N. P., Sugden, T., Hume, A. C., Jackson, M., Krause, A. and Palansuriya, C.: 2004, Introduction to ogsa-dai services, *in* P. Herrero, M. S. Perez and V. Robles (eds), *Proceedings of Scientific Applications of Grid Computing (SAG2004)*, Vol. 3458 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin.

Katsiri, E., Cohen, J., Darlington, J. and Drossopoulou, S.: 2006, Design patterns for grid services, *Proceedings of the Second International Conference "from Scientific Computing to Computational Engineering" (IC_SCCE).*

Keahey, K., Fredian, T., Peng, Q., Schissel, D. P., Thompson, M., Foster, I., Greenwald, M. and McCune, D.: 2002, Computational grids in action: the National Fusion Collaboratory, *Future Generation Computer Science* **18**, 1005–1015.

Kleinrock, L.: 1975, *Queueing Systems Volume 1: Theory*, John Wiley & Sons, New York.

Klienrock, L.: 1969, UCLA press release.
URL: *http://www.lk.cs.ucla.edu/LK/Bib/REPORT/press.html*

Klous, S., Frey, J., Son, S.-C., Thain, D., Roy, A., Livny, M. and van den Brand, J.: 2006, Transparent access to grid resources for user software, *Concurrency and Computation: Practice and Experience* **18**, 787–801.

Knowles, J. D. and Corne, D. W.: 2000, Approximating the nondominated front using the Pareto archived evolution strategy, *Evolutionary Computation* **8**(2), 149–172.

Koza, J. R.: 1992, *Genetic Programming: on the Programming of Computers by Means of Natural Selection*, MIT Press, London.

Koza, J. R., Keane, M. A., Yu, J., Bennett III, F. H. and Mydlowec, W.: 2000, Automatic creation of human-competitive programs and controllers by means of genetic programming, *Genetic Programming and Evolvable Machines* **1**, 121–164.

Koza, J. R., Keane, M. A., Yu, J., III, F. H. B., Mydlowec, W. and Stiffelman, O.: 1999, Automatic synthesis of both the topology and parameters for a robust controller for a non-minimal phase plant and a three-lag plant by means of genetic programming, *Proceedings of 1999 IEEE Conference on Decision and Control*, pp. 5292–5300.

Krasnogor, N. and Smith, J.: 2005, A tutorial for competent memetic algorithms: Model, taxonomy, and design issues, *IEEE Transactions on Evolutionary Computation* **9**(5), 474–488.

Krauter, K., Buyya, R. and Maheswaran, M.: 2002, A taxonomy and survey of grid resource management systems for distributed computing, *Software - Practice and Experience* **32**, 135–164.

Kristinsson, K. and Dumont, G. A.: 1992, System identification and control using genetic algorithms, *IEEE Transactions on Systems, Man, and Cybernetics* **22**(5), 1033–1046.

Kursawe, F.: 1991, A variant of evolution strategies for vector optimization, *in* H.-P. Schwefel and R. Männer (eds), *Parallel Problem Solving from Nature 1*, Vol. 496 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 193–197.

Lahanas, M., Schreibmann, E., Milickovic, N. and Baltas, D.: 2003, Intensity modulated beam radiation therapy dose optimization with multiobjective

evolutionary algorithms, *in* C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb and L. Thiele (eds), *Proceedings of the Second International Conference on Evolutionary Multi-criteria Optimization (EMO) 2003*, Vol. 2632 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 648–661.

Lampinen, J. and Storn, R.: 2004, Differential evolution, *in* G. C. Onwubolu and B. V. Babu (eds), *New Optimization Techniques in Engineering*, Springer, Berlin, pp. 123–166.

Langdon, W. B.: 1995, Scheduling planned maintenance of the national grid, *in* T. C. Fogarty (ed.), *Evolutionary Computing - AISB Workshop*, Vol. 993 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 132–153.

Laumanns, N., Laumanns, M. and Neunzig, D.: 2001, Multi-objective design space exploration of road trains with evolutionary algorithms, *in* E. Zitzler, K. Deb, L. Thiele and C. A. C. C. D. Corne (eds), *Proceedings of the First International Conference on Evolutionary Multi-criteria Optimisation (EMO) 2001*, Vol. 1993 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 612–623.

Lee, D., Lin, A. W., Hutton, T., Akiyamac, T., Shinji, S., Lin, F.-P., Peltier, S. and Ellisman, M. H.: 2003, Global telescience featuring ipv6 at igrid2002, *Future Generation Computer Science* **19**, 1031–1039.

Lenstra, A. K. and Manasse, M. S.: 1989, Factoring by electronic mail, *in* J. J. Quisiquater and J. Vandewalle (eds), *Proceedings of EUROCrypt'89 - Advances in Cryptography*, Vol. 434 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 355–371.

Leslie, R. and McKenzie, S.: 1999, Evaluation of load sharing algorithms for heterogeneous distributed systems, *Computer Communications* **22**, 376–389.

Lewis, R. and Paechter, B.: 2005, Application of the grouping genetic algorithm to university course timetabling, *in* G. R. Raidl and J. Gottlieb (eds), *Proceedings of the Fifth European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP)*, Vol. 3448 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 144–153.

Liang, K.-H., Yao, X. and Newton, C.: 1999, Combining landscape approximation and local search in global optimization, *Proceedings of the Congress on Evolutionary Computation (CEC) 1999*, IEEE Press, pp. 1514–1520.

Licklider, J. R. and Taylor, R. W.: 1968, The computer as a communication device, *Science and Technology* .

Liepins, G. E. and Potter, W. D.: 1991, A genetic algorithm approach to multiple-fault diagnosis, *in* L. Davis (ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, pp. 237–250.

Little, J. D. C.: 1961, A proof for the queueing formula: $L = \lambda w$, *Operations Research* 9(3), 383–387.

Livny, M. and Raman, R.: 1999, High throughput resource management, *in* I. Foster and C. Kesselman (eds), *The GRID: Blueprint for a New Computing Infrastructure*, first edition edn, Morgan Kaufmann, chapter 13, pp. 311–337.

Luke, S.: 1998, Genetic programming produced competitive soccer softbot teams for RoboCup97, *in* J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba and R. Riolo (eds), *Genetic Programming 1998: Proceedings of the Third Annual Conference*, Morgan Kaufmann, pp. 214–222.

Luna, F., Nebro, A. J. and Alba, E.: 2004, A globus-based distributed enumerative search algorithm for multi-objective optimization, *Technical Report LCC 2004/02*, University of Malaga.

Manly, B. F.: 1991, *Randomization and Monte Carlo Methods in Biology*, Chapman and Hall.

Manos, S., Poladian, L., Bentley, P. and Large, M.: 2005, Photonic device design using multiobjective evolutionary algorithms, *in* C. A. C. Coello, A. H. Aguirre and E. Zitzler (eds), *Proceedings of the Third International Conference on Evolutionary Multi-criteria Optimization (EMO) 2005*, Vol. 3410 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 636–650.

Massebeuf, S., Fonteix, C., Kiss, L. N., Marc, I., Pla, F. and Zaras, K.: 1999, Multicriteria optimization and decision engineering of an extrusion process aided by a diploid genetic algorithm, *in* P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao and A. Zalzala (eds), *Proceedings of the Congress on Evolutionary Computation (CEC) 1999*, IEEE Press, pp. 14–21.

McCormick, B. H., DeFanti, T. A. and Brown, M. D.: 1987, Visualization in scientific computing, *Computer Graphics* **21**(6).

McFarlane, D. and Glover, K.: 1990, *Robust Controller Design Using Normalized Coprime Factor Plant Descriptions*, Vol. 138 of *Lecture Notes in Control and Information Sciences*, Springer-Verlag.

MDSplus project: 2006, MDSplus project website. Viewed 28 August 2006.
URL: *http://www.psfc.mit.edu/mdsplus/*

Merz, P. and Freisleben, B.: 1999, A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem, *Proceedings of the Congress on Evolutionary Computation (CEC) 1999*, IEEE Press, pp. 2063–2070.

Mesghouni, K., Hammadi, S. and Borne, P.: 2004, Evolutionary algorithms for job-shop scheduling, *International Journal of Applied Mathematics and Computer Science* **14**(1), 91–103.

Messina, P.: 1999, Distributed supercomputing applications, *in* I. Foster and C. Kessleman (eds), *The GRID: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, chapter 3, pp. 55–73.

Messina, P., Brunett, S., Davis, D. and Gottschalk, T.: 1997, Synthetic forces express: A new initiative in scalable computing for military simulation, *Technical Report cacr136*, California Institute of Technology. Available from `www.cacr.caltech.edu/SFExpress/pubs/cacr136.pdf`.

Michalewicz, Z.: 1996, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin.

Michalewicz, Z. and Fogel, D. B.: 2000, *How to Solve It: Modern Heuristics*, Springer, Berlin.

Michalewicz, Z. and Schoenauer, M.: 1996, Evolutionary algorithms for constrained parameter optimization problems, *Evolutionary Computation* 4(1), 1–32.

Miettinen, K.: 1999, *Nonlinear Multiobjective Optimization*, Kluwer, Boston.

Miller, B. L. and Goldberg, D. E.: 1995, Genetic algorithms, tournament selection, and the effects of noise, *IlliGAL Report 95006*, University of Illinois.

Miller, G. A.: 1956, The magical number seven, plus or minus two: Some limits on our capacity for processing information, *The Psychological Review* **63**(2), 81–97.

Moscato, P.: 1989, On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, *C3P Report 826*, Caltech.

Moscato, P. and Norman, M. G.: 1992, A "memetic" approach for the travelling salesman problem, implementation of a computational ecology for combinatorial optimization on message-passing systems, *in* M. Valero, E. Onate, M. Jane, J. L. Larriba and B. Suarez (eds), *Parallel Computing and Transputer Applications*, IOS Press, Amsterdam, pp. 177–186.

Mühlenbein, H. and Schlierkamp-Voosen, D.: 1993, Predictive models for the breeder genetic algorithm I: Continuous parameter optimization, *Evolutionary Computation* **1**(1), 25–49.

Nelson, R. C.: 1998, *Flight Stability and Automatic Control*, second edition edn, McGraw-Hill.

Ohio Supercomputer Center (OSC): 2007, Cluster Ohio. Viewed 22nd April 2007. URL: *http://www.osc.edu/hpc/cluster_ohio/*

Oliveira, P., Sequeira, J. and Sentieiro, J.: 1991, Selection of controller parameters using genetic algorithms, *in* S. G. Tzafestas (ed.), *Engineering Systems with Intelligence. Concepts, Tools, and Applications*, Kluwer, pp. 431–438.

Ong, M., Ren, X., Allan, G. M., Kadirkamanathan, V., Thompson, H. A. and Fleming, P. J.: 2005, Decision support system on the grid, *International Journal of Knowledge-Based and Intelligent Engineering Systems* **9**, 315–326.

Ong, Y.-S., Lin, M.-H., Zhu, N. and Wong, K.-W.: 2006, Classification of adaptive memetic algorithms: A comparative study, *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics* **36**(1), 141–152.

Parker, S. G., Johnson, C. R. and Beazley, D.: 1997, Computational steering software systems and strategies, *IEEE Computational Science and Engineering* 4(4), 50–59.

Parmee, I.: 2002, Improving problem definition through interactive evolutionary computation, *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing* **16**(3), 185–202.

Parmee, I. C., Cvetkovic, D., Watson, A. H. and Bonham, C. R.: 2000, Multiobjective satisfaction within an interactive evolutionary design environment, *Evolutionary Computation* 8(2), 197–222.

Piccolboni, A. and Mauri, G.: 1998, Application of evolutionary algorithms to protein folding prediction, *in* J.-K. Hao, E. Lutton, E. M. A. Ronald, M. Schoenauer and D. Snyers (eds), *Proceedings of the Third European Conference on Artificial Evolution (EA 1997)*, Vol. 1363 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 123–136.

Platform Computing Corporation: 2002, Platform LSF user's guide.

Platform Computing Corporation: 2007, Platform LSF product page. Viewed 22 March 2007.
URL: *http://www.platform.com/Products/Platform.LSF.Family/home.htm*

Polya, G.: 1957, *How to Solve It: A New Aspect of Mathematical Method*, second edition edn, Doubleday, New York.

Porter, B. and Jones, A. H.: 1992, Genetic tuning of digital PID controllers, *Electronics Letters* **28**(9), 843–844.

Price, K. and Storn, R.: 1995, Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces, *Technical*

Report Technical Report TR-95-012, International Computer Science Institute, Berkeley.

Purshouse, R. C.: 2003, *On the Evolutionary Optimisation of Many Objectives*, PhD thesis, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK, S1 3JD.

Raman, R., Livny, M. and Solomon, M.: 2000, Resource management through multilateral matchmaking, *Proceedings of the Ninth IEEE Symposium on High Performance Distributed Computing (HPDC9)*, pp. 290–291.

Rasheed, K. and Hirsh, H.: 2000, Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models, *Proccedings of the Genetic and Evolutionary Computation Conference (GECCO) 2000*, Morgan Kaufmann, pp. 628–635.

Rasheed, K., Ni, X. and Vattam, S.: 2003, Comparison of methods for developing dynamic reduced models for design optimization, *Proceedings of the Congress on Evolutionary Computation (CEC) 2002*, IEEE Press, pp. 390–395.

Rechenberg, I.: 1973, *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*, Frommann-Holzboog, Stuttgart.

Regis, R. G. and Shoemaker, C. A.: 2004, Local function approximation in evolutionary algorithms for the optimization of costly functions, *IEEE Transactions on Evolutionary Computation* **8**(5), 490–505.

Rekiek, B., Lit, P. D., Fabrice, P., L'Eglise, T., Emanuel, F. and Delchambre, A.: 2000, Dealing with user's preferences in hybrid assembly lines design, *Proceedings of MCPL'2000.*

Rivera, W.: 2001, Scalable parallel genetic algorithms, *Artificial Intelligence Review* **16**(2), 153–168.

Rizki, M. M., Zmuda, M. A. and Tamburino, L. A.: 2002, Evolving pattern recognition systems, *IEEE Transactions on Evolutionary Computation* **6**(6), 594–609.

Rolls-Royce: 1986, *The Jet Engine*, 4th edition edn, Rolls Royce, Derby, UK.

Rolls-Royce: 2002, Mearos model description version 8.31, Rolls-Royce Internal Document.

Rudolph, G. and Agapie, A.: 2000, Convergence properties of some multi-objective evolutionary algorithms, *Proceedings of the Congress on Evolutionary Computation (CEC) 2000*, IEEE Press, pp. 1010–1016.

Rumbaugh, J., Jacobson, I. and Booch, G.: 1999, *The Unified Modeling Language Reference Manual*, Addison-Wesley, Reading, MA.

Sano, Y. and Kita, H.: 2000, Optimization of noisy fitness functions by means of genetic algorithms using history of search, *in* M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo and H.-P. Schwefel (eds), *Parallel Problem Solving from Nature VI*, Vol. 1917 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 360–365.

Schaffer, J. D.: 1985, Multiple objective optimization with vector evaluated genetic algorithms, *in* J. J. Grefenstette (ed.), *Proceedings of the First International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, New Jersey, pp. 93–100.

Schissel, D. P.: 2005, Grid computing and collaboration technology in support of fusion energy sciences, *Physics of Plasmas* **12**.

Schissel, D. P., Keahey, K., Araki, T., Burruss, J. R., Feibush, E., Flanagan, S. M., Foster, I., Fredian, T. W., Greenwald, M. J., Klasky, S. A., Leggett, T., Li, K., McCune, D. C., Lane, P., Papka, M. E., Peng, Q., Randerson,

L., Sanderson, A., Stillerman, J., Thompson, M. R. and Wallace, G.: 2004, The National Fusion Collaboratory project: Applying grid technology for magnetic fusion research, *Proceedings of the Workshop on Case Studies on Grid Applications at GGF10.*

Schlottmann, F., Mitschele, A. and Seese, D.: 2005, A multi-objective approach to integrated risk management, *in* C. A. C. Coello, A. H. Aguirre and E. Zitzler (eds), *Proceedings of the Third International Conference on Evolutionary Multi-criteria Optimization (EMO) 2005*, Vol. 3410 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 692–706.

Schopf, J. M.: 2002, A general architecture for scheduling on the grid, *Technical Report ANL/MCS-P1000-1002*, Argonne National Laboratory.

Schopf, J. M., D'Arcy, M., Miller, N., Pearlman, L., Foster, I. and Kesselman, C.: 2005, Monitoring and discovery in a web services framework: Functionality and performance of the Globus Toolkit's MDS4, *Argonne National Laboratory Technical Report ANL/MCS-P1248-0405*, Argonne National Laboratory.

Schroder, P.: 1998, *Multivariable Control of Active Magnetic Bearings*, PhD thesis, University of Sheffield.

Schulze-Kremer, S.: 1992, Genetic algorithms for protein tertiary structure generation, *in* R. Männer and B. Manderick (eds), *Parallel Problem Solving from Nature 2*, North-Holland, pp. 393–403.

SCIRun: 2006, Scirun: A scientific computing problem solving environment. http://software.sci.utah.edu/scirun.html.

Segaller, S.: 1998, *Nerds: A Brief History of the Internet*, T.V. Books.

SETI@home project: 2006, SETI@home website. Viewed 28 August 2006.
URL: *http://setiathome.berkeley.edu/*

Shenfield, A. and Fleming, P. J.: 2005, A service oriented architecture for decision making in engineering design, *in* P. M. A. Sloot, A. G. Hoekstra, T. Priol, A. Reinefeld and M. Bubak (eds), *Advances in Grid Computing: EGC 2005*, Vol. 3470 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 334–343.

Shenfield, A., Fleming, P. J. and Alkarouri, M.: 2007, Computational steering of a multi-objective evolutionary algorithm for engineering design, *Engineering Applications of Artificial Intelligence* . (In Press) doi: 10.1016/j.engappai.2007.01.005.

Shenfield, A., Fleming, P. J., Allan, J. and Kadirkamanathan, V.: 2007, Optimisation of maintenance scheduling strategies on the grid, *Proceedings of the Symposium Series on Computational Intelligence (SSCI) 2007.*

Shenfield, A., Ong, M., Allan, G. M., Ren, X., Kadirkamanathan, V., Thompson, H. A. and Fleming, P. J.: 2005, Modelling, optimisation and decision support using the grid, *in* S. J. Cox and D. W. Walker (eds), *Proceedings of the UK e-Science All Hands Meeting (AHM) 2005.*

Silverman, B. W.: 1986, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London.

Simpson, T. W., Mauery, T. M., Korte, J. J. and Mistree, F.: 1998, Comparison of response surface and kriging models for multidisciplinary design optimization, *Technical Report AIAA-98-4755*, American Institute of Aeronautics and Astronautics.

Sims, K.: 1991, Artificial evolution for computer graphics, *Computer Graphics* **25**(4), 319–328.

Skogestad, S. and Postlethwaite, I.: 1996, *Multivariable feedback control - Analysis and Design*, John Wiley & Sons.

Smarr, L. and Catlett, C. E.: 1992, Metacomputing, *Communications of the ACM* **35**(6), 45–52.

Song, W., Ong, Y. S., Ng, H. K., Keane, A., Cox, S. and Lee, B. S.: 2004, A service-oriented approach for aerodynamic shape optimization across institutional boundaries, *Proceedings of the 8th ICARCV Control, Automation, Robotics and Vision Conference*, pp. 2274–2279.

Srinivas, N. and Deb, K.: 1994, Multiobjective optimization using nondominated sorting in genetic algorithms, *Evolutionary Computation* **2**(3), 221–248.

Starkweather, T., Whitley, D. and Mathias, K.: 1991, Optimization using distributed genetic algorithms, *in* H.-P. Schwefel and R. Männer (eds), *Parallel Problem Solving from Nature 1*, Vol. 496 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 176–185.

Suganathan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A. and Tiwari, S.: 2005, Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization, *Technical Report 2005005*, KanGAL.

Sun Microsystems Inc.: 2006, N1 grid engine 6 user's guide. Viewed 22 March 2007.
URL: *http://gridengine.sunsource.net/documentation.html*

Tabak, D., Schy, A. A., Giesy, D. P. and Johnson, K. G.: 1979, Application of multiobjective optimization in aircraft control system design, *Automatica* **15**, 595–600.

Takagi, H.: 2001, Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation, *Proceedings of the IEEE* **89**(9), 1275–1296.

Tan, K. C., Tay, A. and Cai, J.: 2003, Design and implementation of a distributed evolutionary computing software, *IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews* **33**(3), 325–338.

Tan, K. C., Yu, Q. and Lee, T. H.: 2005, A distributed evolutionary classifier for knowledge discovery in data mining, *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews* **35**(2), 131–142.

Tanese, R.: 1987, Parallel genetic algorithms for a hypercube, *Proceedings of the Second International Conference on Genetic Algorithms (ICGA2)*, pp. 177–183.

Tang, X. and Chanson, S. T.: 2000, Optimizing static job scheduling in a network of heterogeneous computers, *Proceedings of the International Conference on Parallel Processing*, pp. 373–382.

Tanimura, Y., Hiroyasu, T., Miki, M. and Aoi, K.: 2002, The system for evolutionary computing on the computational grid, *Proceedings of the 14th International Conference on Parallel and Distributed Computing Systems*, ACTA press, pp. 39–44.

Tannenbaum, A. S.: 1996, *Computer Networks*, third edn, Prentice Hall, New Jersey.

Thain, D., Tannenbaum, T. and Livny, M.: 2003, Condor and the grid, *in* F. Berman, G. Cox and A. Hey (eds), *Grid Computing - Making the Global Infrastructure a Reality*, John Wiley & Sons.

The Beowulf Project: 2006, Beowulf Project website. Viewed 28 August 2006.
URL: *http://www.beowulf.org/*

The Condor Project: 2007, Condor project website. Viewed 22 March 2007.
URL: *http://www.cs.wisc.edu/condor/*

The European DataGrid Project: 2007, The dataGrid project website. Viewed 22 March 2007.

URL: *http://eu-datagrid.web.cern.ch/eu-datagrid/*

The Globus Alliance: 2006, Globus Project website. Viewed 28 August 2006.

URL: *http://www.globus.org/*

The Globus Security Team: 2005, Globus Toolkit version 4 Grid Security Infrastructure: A standards perspective, *Technical report*, The Globus Project. Available from `http://www.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf`.

The GridPP Project: 2007, GridPP - uk computing for particle physics website. Viewed 23 March 2007.

URL: *http://www.gridpp.ac.uk/*

The International Human Genome Sequencing Consortium: 2004, Finishing the euchromatic sequence of the human genome, *Nature* **431**, 931–945.

The National Fusion Collaboratory Project: 2001, National Fusion Collaboratory: Executive summary, Website.

URL: *http://www.fusiongrid.org/about/exec-sum.html*

The Sun Grid Engine Project: 2007, Sun grid engine project website. Viewed 22 March 2007.

URL: *http://gridengine.sunsource.net/*

The White Rose University Consortium: 2007, The white rose grid website. Viewed 23 March 2007.

URL: *http://www.wrgrid.org.uk/index.html*

Tierney, B., Aydt, R., Gunter, D., Smith, W., Taylor, V., Wolski, R. and Swany, M.: 2002, A grid monitoring architecture, *Technical Report GWD-Perf-16-3*, Global Grid Forum (Performance Working Group).

Todd, D. S. and Sen, P.: 1999, Directed multiple objective search of design spaces using genetic algorithms and neural networks, *in* W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela and R. E. Smith (eds), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 1999*, Morgan Kaufmann, pp. 1738–1743.

Top500 List: 2006, Released at the International Supercomputer Conference 2006.
URL: *http://www.top500.org/lists/2006/06*

Tuecke, S.: 2001, Grid security infrastructure roadmap, *Technical report*, Global Grid Forum (GSI Working Group). Available from `http://www.gridforum.org/security/ggf1_2001-03/drafts/draft-ggf-gsi-roadmap-02.txt`.

Tzafestas, S. G., Saltouros, M.-P. and Markaki, M.: 1999, A tutorial overview of genetic algorithms and their applications, *in* S. G. Tzafestas (ed.), *Soft Computing in Systems and Control Technology*, World Scientific, New Jersey, pp. 223–300.

Ulmer, H., Streichert, F. and Zell, A.: 2003, Evolution strategies assisted by gaussian processes with improved pre-selection criterion, *Proceedings of the Congress on Evolutionary Computation (CEC) 2003*, IEEE Press, pp. 692–699.

United Devices: 2004, Human proteome folding project website. Viewed 28 August 2006.
URL: *http://www.grid.org/projects/hpf/*

Vadhiyar, S. S. and Dongarra, J. J.: 2002, A metascheduler for the grid, *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11)*, pp. 343–351.

Van Veldhuizen, D. A.: 1999a, *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*, PhD thesis, Air Force Institute of Technology.

Van Veldhuizen, D. A.: 1999b, *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*, PhD thesis, Air Force Institute of Technology.

Van Veldhuizen, D. A. and Lamont, G. B.: 2000, On measuring multiobjective evolutionary algorithm performance, *Proceedings of the Congress on Evolutionary Computation (CEC) 2000*, IEEE Press, pp. 204–211.

Van Veldhuizen, D. A., Zydallis, J. B. and Lamont, G. B.: 2003, Considerations in engineering parallel multi-objective evolutionary algorithms, *IEEE Transactions on Evolutionary Computation* 7(2), 144–173.

Varcol, M. C. and Emmerich, M.: 2005, Metamodel-assisted evolution strategies applied in electromagnetics, *in* R. Schilling, W. Haase, J. P'eriaux and H. Baier (eds), *Proceedings of the European Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN) 2005.*

Vlachos, C., Williams, D. and Gomm, J. B.: 1999, Genetic approach to decentralised PI controller tuning for multivariable processes, *IEE Proceedings - Control Theory and Applications*, Vol. 146, pp. 58–64.

Vyssotsky, V. A., Corbató, F. J. and Graham, R. M.: 1965, Structure of the multics supervisor, *Proceedings of the Fall Joint Computer Conference*, Vol. 27. Available from `http://www.multicians.org/fjcc3.html`.

W3C Working Group: 2004, Web services architecture. Viewed 18 October 2006.
URL: *http://www.w3c.org/TR/ws-arch*

Wang, Y. and Morris, R.: 1985, Load sharing in distributed systems, *IEEE Transactions on Computers* **34**(3), 204–217.

Wegman, E. J.: 1990, Hyperdimensional data analysis using parallel coordinates, *Journal of the American Statistical Association* **85**(411), 664–675.

White, C. M. and Yen, G. G.: 2004, A hybrid evolutionary algorithm for travelling salesman problem, *Proceedings of the Congress on Evolutionary Computation (CEC) 2004*, IEEE Press, pp. 1473–1478.

White III, C. C., Sage, A. P. and Dozono, S.: 1984, A model of multiattribute decision making and trade-off weight determination under uncertainty, *IEEE Transactions on Systems, Man, and Cybernetics* **14**(2), 223–229.

Whitley, D.: 1989, The GENITOR algorithm and selection pressure: Why rank based allocation of reproductive trials is best, *in* J. D. Schaffer (ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 116–121.

Wienke, D., Lucasius, C. and Kateman, G.: 1992, Multicriteria target vector optimization of analytical procedures using a genetic algorithm - part I. theory, numerical simulations and application to atomic emission spectroscopy, *Analytica Chimica Acta* **265**, 211–225.

Wierzbicki, A. P.: 1980, The use of reference objectives in multiobjective optimization, *in* G. Fandel and T. Gal (eds), *Multiple Criteria Decision Making Theory and Applications*, Springer-Verlag, Berlin, pp. 468–486.

Wolski, R.: 1996, Forecasting network performance to support dynamic scheduling using the network weather service, *Proceedings of the sixth IEEE Symposium on High Performance Distributed Computing (HPDC6)*, pp. 316–325.

Wright, S.: 1932, The roles of mutation, inbreeding, crossbreeding, and selection in evolution, *Proceedings of the 6th International Conference of Genetics.*

Yarmolenko, V. and Sakellariou, R.: 2007, Towards increased expressiveness in service level agreements, *Concurrency and Computation: Practice and Experience* . (In Press) doi: 10.1002/cpe.1140.

Yu, P. L.: 1973, A class of solutions for group decision problems, *Management Science* **19**(8), 936–946.

Zames, G.: 1981, Feedback and optimal sensitivity: model reference transformations, multiplicitive seminorms, and approximate inverse, *IEEE Transactions on Automatic Control* **26**(2), 301–320.

Zitzler, E., Deb, K. and Thiele, L.: 2000, Comparison of multiobjective evolutionary algorithms: Empirical results, *Evolutionary Computation* **8**(2), 173–195.

Zitzler, E., Laumanns, M. and Bleuler, S.: 2004, A tutorial on evolutionary multiobjective optimization, *in* X. Gandibleux, M. Sevaux, K. Sörensen and V. T'Kindt (eds), *Methaheuristics for Multiobjective Optimisation*, Vol. 535 of *Lecture Notes in Economics and Mathematical Systems*, Springer-Verlag, Berlin.

Zitzler, E., Laumanns, M. and Thiele, L.: 2001, SPEA2: Improving the strength Pareto evolutionary algorithm, *TIK Report 103*, TIK Institut fur Technische Informatik und Kommunikationsnetze, Computer Engineering and Networks Laboratory, ETH, Swiss Federal Institute of Technology, Gloriastrasse 35, 8092 Zurich, Switzerland.

Zitzler, E. and Thiele, L.: 1999, Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach, *IEEE Transactions on Evolutionary Computation* **3**(4), 257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M. and da Fonseca, V. G.: 2003, Performance assessment of multiobjective optimizers: An analysis and review, *IEEE Transactions on Evolutionary Computation* **7**(2), 117–132.